$9.95

SOUND AND GRAPHICS

For the APPLE II+, //e and FRANKLIN COMPUTERS

by Jerry Abad and Valerie Abad

Learn BASIC graphics techniques in colorful and musical ways!

SOUND AND GRAPHICS

SOUND AND GRAPHICS

For the APPLE II+, //e and FRANKLIN COMPUTERS

by

Jerry Abad

and

Valerie Abad

Illustrated by
Martin Cannon

Technical Drawings by
Tes Bonnarith

DATAMOST™

20660 Nordhoff Street
Chatsworth, CA 91311-6152
(818) 709-1202

RESTON PUBLISHING COMPANY, INC.
A Prentice-Hall Company
Reston, Virginia

ISBN 0-8359-7055-8

Copyright © 1984 DATAMOST Inc.
All Rights Reserved

This manual is published and copyrighted by DATAMOST Inc. Copying, duplicating, selling or otherwise distributing this product is hereby expressly forbidden except by prior written consent of DATAMOST Inc.

The Apple and the Apple logo are registered trademarks of Apple Computer Inc.

Apple Computer Inc. was not in any way involved in the writing or other preparation of this book, nor were the facts presented here reviewed for accuracy by that company. Use of the term Apple should not in any way be construed to represent any endorsement, official or otherwise, by Apple Computer Inc.

Printed in U.S.A.

Dedicated to the memory of Jerry's brother, Rick Abad

and to

Valerie's loving parents, Mildred and Bob Jones

Table of Contents

Preface 9
Programming Tips and Techniques 11
Low Resolution Graphics 19
GOTO Loops — IF/THEN 25
VLIN and HLIN 29
Area of a Rectangle in Graphics 31
Background Colors 33
Moving Graphics 35
High Resolution Graphics 39
Three Dimensional Graphics 43
A-Mazing Graphics 51
Hi-Res Background Colors 55
Hi-Res Action 57
Alternating Graphics Pages One and Two 59
Drawing in Hi-Res with Paddles 63
Using Joystick/Paddles to Fire Missiles 65
Shooting Random Targets 67
U.F.O. 69
Close Encounters 71
Sound Effects for Games 73
Sound Subroutines 77
Introduction to Apple Music 81
Space Music 85
Combining Graphics and Sound 87
Pictures and Shape Tables 91
Creating Shape Tables 95
Answer Key 107

Preface

As the directors of DATA BASE COMPUTER CAMP, we have developed an extensive computer curriculum. The lessons in this book evolved from some of our best experiences while teaching graphics and sound programming during several years of summer camps. In the process of compiling useful programs for instruction, we were struck by the lack of "user-friendly" books on graphics techniques. There were virtually no books that used a building block approach to teach graphic skills on the computer.

The more we taught, the more we realized that there was a great need for material on programming graphics and sound. Together we decided it was necessary to write a book that would teach beginning through advanced graphics techniques to both children and adults. While we were preparing the manuscript we decided that sound routines would be a great addition. Thus was born *Sound and Graphics*.

Jerry and Valerie Abad
Oakland, California
March, 1984

Note

All of these programs have been tested on the Apple II+, Apple //e, and the Franklin computers. When programming on the Apple //e, you must keep the shift lock key down since commands in lower case are not understood on the Apple //e.

Programming Tips and Techniques

This section is designed for people who have limited programming experience in BASIC. If you have programmed in BASIC before, you may wish to skim this chapter or skip to the next chapter and begin graphics programming. We will outline BASIC commands and techniques necessary to begin writing graphics programs in this chapter.

Commands

Useful commands and ideas which we will discuss in this chapter are:

LIST	PLOT	SAVE
GR	LOAD	RUN
TEXT	HOME	NEW
COLOR	GOTO	FOR/NEXT loop
IF/THEN	REM	: (colon)

Glossary of Commands

LIST — This command will tell the computer to show the lines of a program.

GR — Sets the monitor screen for lo-res graphics.

TEXT — Resets the monitor screen to accept standard text type letters and characters.

COLOR — Sets a color for a graphics program. You must set a color even on a monochrome (one color) screen, or the computer will draw everything in the same color as the screen.

IF/THEN — Conditional statement. Lets the computer evaluate conditions in the program as it runs.

PLOT — The command to color in squares on a low resolution graphics program.

LOAD — Allows you to load a program into memory without running it. This can be helpful if you don't want to run through the whole program, but just want to list it. Just type LOAD and the program name. Then, when the cursor appears, type LIST.

COLON — The colon (:) can be used to separate different commands placed on the same line. For example, 10 HOME : PRINT "HI" will clear the screen and print the word "HI" at the top.

HOME — Clears the screen and returns the cursor to the top. Does not clear the memory.

GOTO — Tells the computer to jump to a different part of the program. For example, 40 GOTO 100, will send the program to line 100.

REM — Stands for remark. Allows the programmer to put information inside the program that isn't a part of the program, but can be used to explain parts of it that might be unclear later on. For example, when you use CTRL-G to make bell sound, nothing shows up on the screen. You could use a REM statement to indicate what you put in the program. It might look like this:

```
1Ø PRINT " " : REM CTRL-G BELL SOUND
```

SAVE — Saves a program onto a disk or tape. You must use it with a program name. For example, SAVE PICTURE will save a program called "PICTURE".

RUN — Makes the computer follow the steps of a program and do what the program says.

NEW — Clears the memory of previous programs. Whenever starting a new program, it is advisable to type NEW first so your program doesn't overlap with a previous program.

FOR/NEXT LOOP — Can be used to make the computer repeat a section of the program as many times as you want.

Writing a Program

A program is a set of instructions for the computer to execute. In the BASIC programming language, the program steps are always numbered. Generally it is not advisable to number the steps with consecutive numbers (1, 2, 3, etc.) because then you cannot put in extra steps if you want to add something to your program. The computer does not understand a line number of 2.5 or 2½. Thus a programmer generally numbers the programs in steps of ten (10, 20, 30 etc.). This way instructions can be inserted at lines 11, 12, 13, 14 or wherever necessary.

Preparing Blank Disks

You will need several blank initialized disks to store the programs in this book. Just follow these steps to get working quickly. First boot up your system master and type in the following as the HELLO program.

```
NEW
10 PRINT "THIS DISK BELONGS TO (put your name here)"
20 PRINT "date, system size, DOS ver."
30 PRINT "title of disk"
```

Here is an example:

```
NEW
10 PRINT "THIS DISK BELONGS TO JOHN ADAMS."
20 PRINT "JAN 1984, 64K SYSTEM, DOS 3.3"
30 PRINT "GRAPHICS PROGRAMS"
```

Now insert a blank disk in the drive and type the following after the prompt:

```
INIT HELLO
```

The drive light will come on for a minute or two, make some noise and prepare the disk. If you want to check whether everything was done correctly, reboot the disk by turning the computer off and then on again. You can also reboot the disk by typing:

```
PR#6
```

You should see the following on the screen:

```
THIS DISK BELONGS TO JOHN ADAMS
JAN 1984, 64K SYSTEM, DOS 3.3
GRAPHICS PROGRAMS
```

If you type LIST then you should see the HELLO program as shown above.

If you type CATALOG you should see the following:

```
DISK VOLUME 254
A 002 HELLO
```

Your HELLO program will be the first program on the disk. The 002 indicates that it takes up 2 sectors of space, and the A indicates that the program is in Applesoft.

Now you should be ready to initialize your own disks by following the above directions. Remember to type NEW to clear out the HELLO program before beginning.

Titles

Many programs begin with a title page. You will probably want a title page when you put your program together. The following program could also be used as a HELLO program. After you create your HELLO program and get it working just save it by typing "SAVE HELLO".

Let's suppose that you want to see the following on the screen:

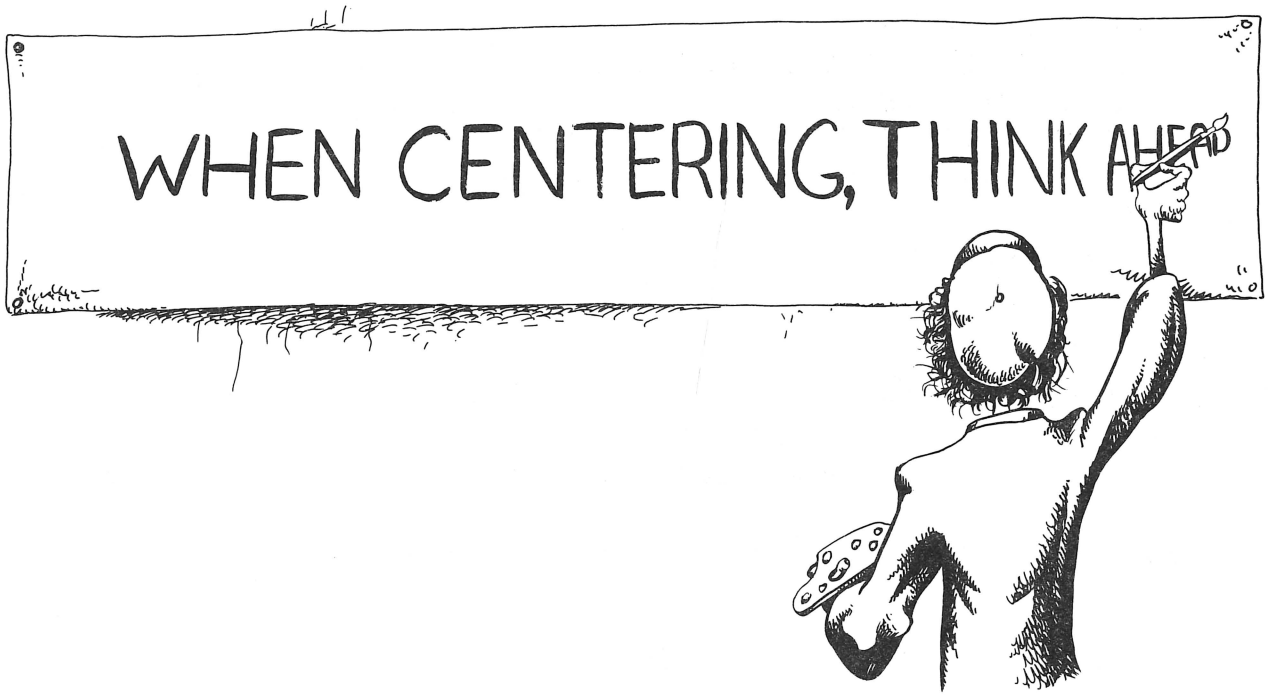
```
SOUND AND GRAPHICS

FOR THE APPLE

BY

JERRY & VALERIE ABAD
```

The screen is 40 columns wide and 24 rows deep. Since there are four lines of text on the title page we want to create, each line could be spaced six rows apart, on rows 3,9,15 and 21. Centering a line is harder, but the concept is simple. Just put the same number of characters to the left of center as you do to the right. If a line is 24 characters long that would mean putting 12 characters on each side of the center point. Since there are 20 spaces on each side of the center point and you need 12 for each side of the title you'll need 8 blank spaces.



The commands you will need are simple: VTAB nn and HTAB nn. VTAB stands for vertical tab and HTAB stands for horizontal tab. The nn represents the starting row or column, and is specified by the user.

For example, VTAB 3 comes down 3 rows from the top and HTAB 11 goes over 11 columns from the left of the screen. SOUND AND GRAPHICS takes up 18 columns and because 18/2=9, and 20-9=11 we will use the following values:

```
20 VTAB 3: HTAB 11
30 PRINT "SOUND AND GRAPHICS"
```

Here is the entire program:

```
10 HOME
20 VTAB 3: HTAB 11
30 PRINT "SOUND AND GRAPHICS"
40 VTAB 9: HTAB 13
50 PRINT "FOR THE APPLE"
60 VTAB 15: HTAB 19
70 PRINT "BY"
80 VTAB 21: HTAB 10
90 PRINT "JERRY & VALERIE ABAD"
```

Challenge Problems

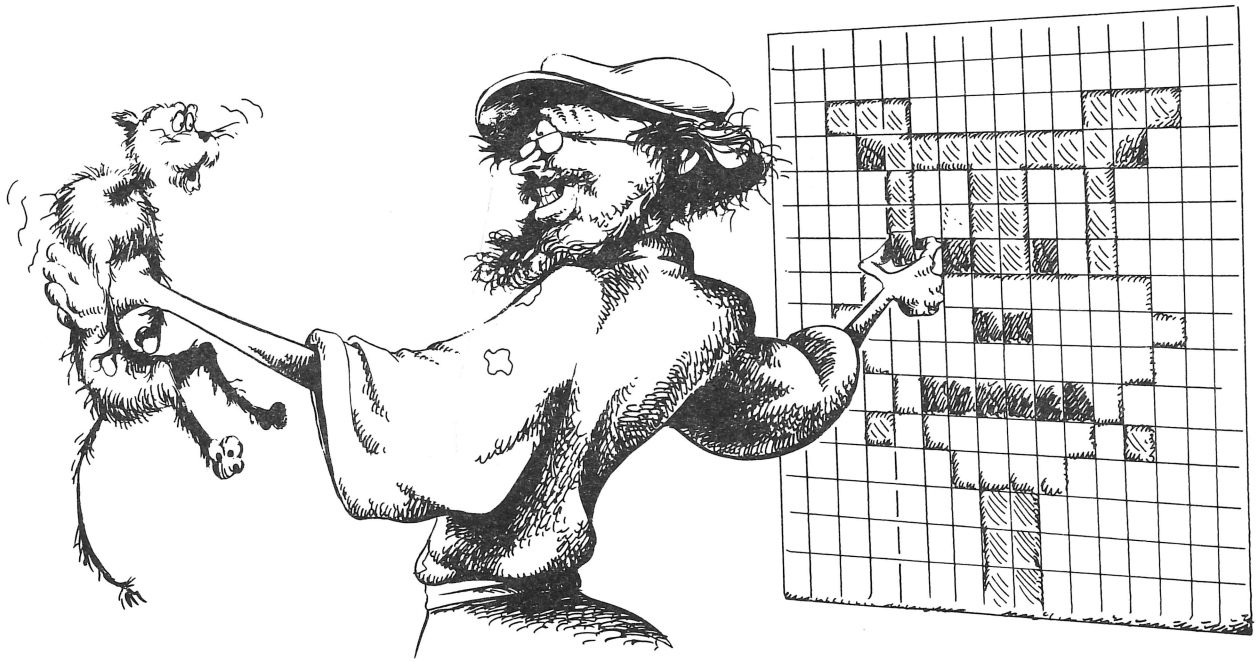
1. Suppose a line contains 14 letters and spaces. What would the HTAB number be?

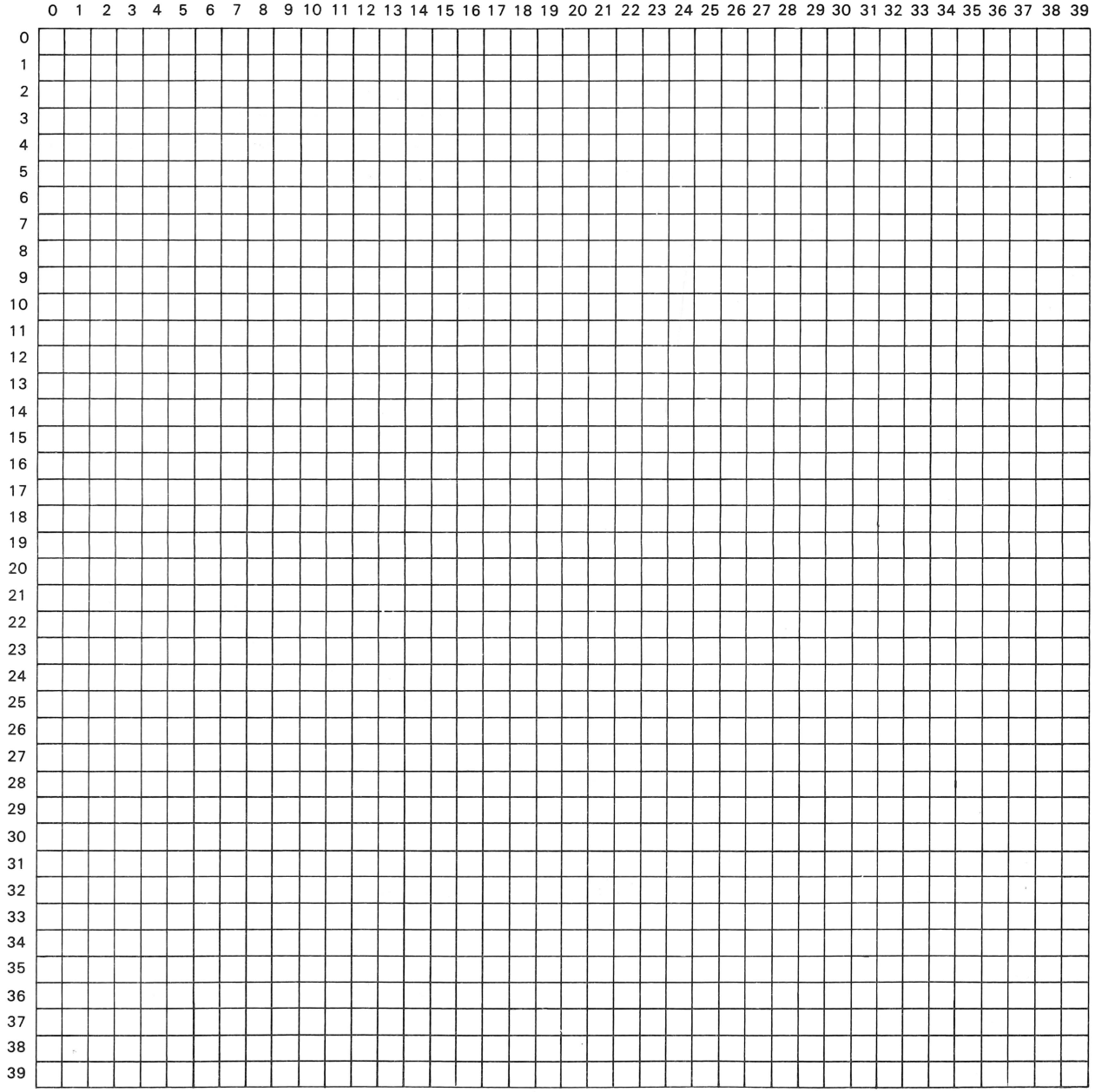
2. Create your own title page and save it as your HELLO program.

Low Resolution Graphics

In low resolution graphics, the monitor screen is divided into a 40 x 40 grid, much like a piece of graph paper. In order to draw in lo-res graphics we must make a block design by coloring in squares to make the desired shape. There are sheets of graph paper at the back of the book to help you design your drawings.

To use the plot command you plot the horizontal (x-axis), then the vertical (y-axis). Remember that in computerese you start counting at 0 so the numbers will go from 0 to 39 on each axis.




0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

Starting at Square One

First, let's make a few squares on screen. We must tell the computer three things in order to color in the squares. We use the GR command to let the computer know that we will be working in the graphics mode. The first line of your program will read:

```
10 GR
```

The second thing we must let the computer know is what color to draw in, so line 20 should read:

```
20 COLOR= 6
```

You may choose any number for 0-15, but remember that if the color you select is black (COLOR= 0) it won't show on the screen because the screen is normally black.

The third command tells the computer where to color in the square. The location is given by two numbers, for example 5,20. The first number is the the number of spaces horizontally from the left side of the screen and the second number indicates the number of rows vertically from the top of the screen. So to color a spot near the left center of the screen you would use the PLOT command as follows:

```
30 PLOT 5,20
```

Once this program has been typed in and the RUN command has been used the computer will remain in graphics mode until the TEXT command is used. TEXT clears the screen of graphics and allows you to see your program again. You may type TEXT without a line number followed by LIST. LIST tells the computer to show the list of the commands in your program. A short way of doing this is by typing:

```
TEXT:LIST
```

Center

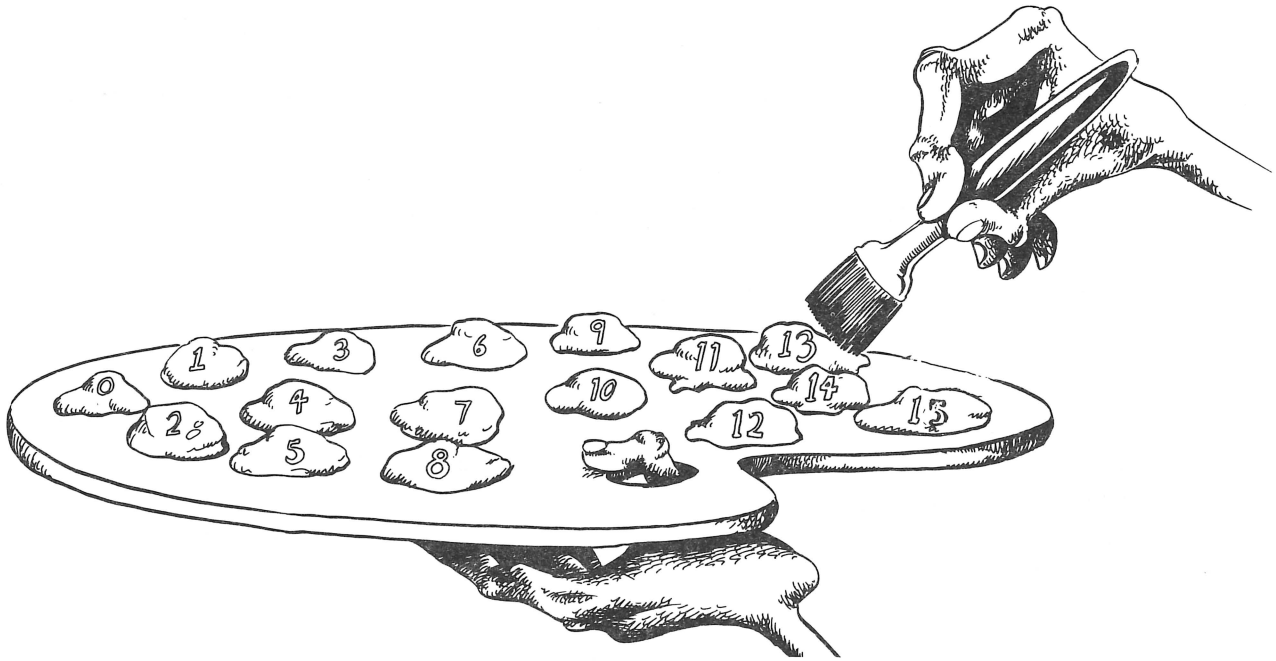
Type in the following program to place a spot in the center of the screen.

```
10 GR
20 COLOR= 9
30 PLOT 20,20
```

Table of Low Resolution Colors

0 black	4 dark green	8 brown	12 light green
1 magenta	5 grey 1	9 orange	13 yellow
2 dark blue	6 medium blue	10 grey 2	14 aquamarine
3 purple	7 light blue	11 pink	15 white

Note: These colors may vary according to the monitor or TV used.



Challenge Programs

1. Change the location of the spot to the lower right part of the screen.

2. Change the color of the spot to yellow.

GOTO Loops — IF/THEN

Type this simple graphics program to see what it does. If you already have a program in memory be sure to type NEW to clear memory before you start typing this program.

```
10 GR : COLOR= 6
20 B = 1
30 C = 1
40 PLOT B,C
50 B = B + 1:C = C + 1
60 IF B = 39 THEN END
70 GOTO 40
```

Now type RUN to tell the computer to execute the program instructions or commands. You should have a series of PLOTs from the upper left part of the screen to the lower right. If it didn't work check to see that you copied the program exactly and that you spelled everything correctly.

Step by Step Through GOTO Loops

Let's look at each part of the program to see how it works. Line 10 says 10 GR : COLOR= 6. The GR tells the computer to set up the screen for doing graphics. The second part, COLOR= 6, tells the computer what color to draw the plots in. Remember that in lo-res graphics there are 16 colors numbered 0-15. If you don't select a color, even if you are using a black and white TV or a green screen monitor you will not see anything, because the computer will draw in the same color as the screen. This will make the design blend in with the background.

When you use the PLOT command you must use two numbers, just as if you were plotting a square on a piece of graph paper. So in lines 20 and 30 we tell the computer to start at 1 for the

first value, B, and at 1 for the second value, C. We are using variables (the letters B and C in this case) to stand for the numbers. This allows us to tell the computer to change the position of the plot each time it executes the loop. Initially B = 1 and C = 1. In line 40 we tell the computer to PLOT B,C which is in the upper left corner of the screen. In line 50 we tell the computer to add 1 to B and 1 to C. This then makes B=2, (1+1) and C=2, (1+1). In line 60 we tell the computer to see IF the number for B equals 39, which is the end of graphics screen. If it is, THEN the program ENDs.

In line 70 we tell the computer to go back to line 40 and execute everything from that point on again. The GOTO command allows the computer to jump forward or backward, skipping to the correct place in the program. Each time the computer executes the loop — goes around the circle from line 70 to line 40 and down to line 70 again and back to line 40 etc. — the computer will add one to B and C because of the command in line 50. It will PLOT 3,3 and then PLOT 4,4 and so forth until it reaches PLOT 39,39. It stops at 39 because of the command in line 60:

```
60 IF B = 39 THEN END
```

Once you have run a graphics program and you want to see a list of your program steps again, you must first type:

```
TEXT
```

This brings you out of the graphics mode and allows you to enter commands. Otherwise, when you list your program all you will see are the last two lines of your program after the rest of it scrolls up "under" the picture. If you just type TEXT the screen will change to various text symbols. If you want to clear the whole screen before LISTing your program, type the following:

```
TEXT <RETURN>
HOME:LIST
```

The colon (:) allows the programmer to put more than one instruction on a line at a time. Just separate each instruction with a colon.

If you want to keep a copy of the program, place a disk that has already been initialized in the drive and type:

```
SAVE PLOTS
```

We chose PLOTS as the name of the file, but you may use any name you like. It's important to remember that you must have a name for your file. Just typing SAVE and hitting return will not work. All programs that you want to save on disk must have a name.

Challenge Problems

1. Use the PLOTS program in this section and see what happens when you change the starting points for B and C.

2. If you have a color monitor try changing the color of the plots.

3. Make the computer print every second plot.

4. Have the line stop in the center of the screen.

5. Add the following line to your program and explain what happens.

```
45 PRINT CHR$(7)
```

VLIN and HLIN

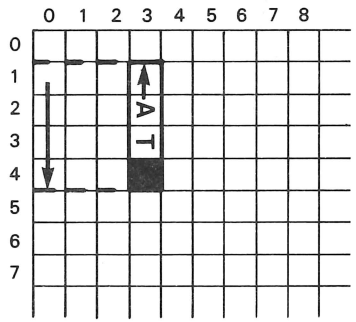
Beyond Plots

Using the PLOT command to draw a line across the screen would be long and tedious. The VLIN and HLIN commmands allow the programmer to draw lines on the screen in a vertical or horizontal position. VLIN stands for vertical line and HLIN stands for horizontal line. The following program draws a cross on the screen:

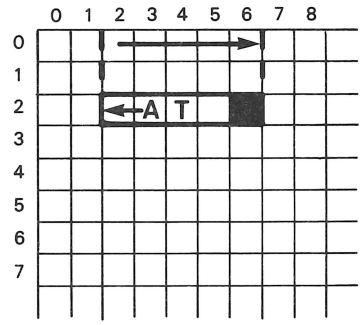
```
10 GR
20 COLOR = 5
30 VLIN 1,39 AT 19
40 HLIN 14,24 AT 15
```

A vertical or horizontal line is written as a series of numbers: where it starts, where it ends and the row or column where it is drawn. For example:

VLIN 1,4 AT 3



HLIN 2,6 AT 2

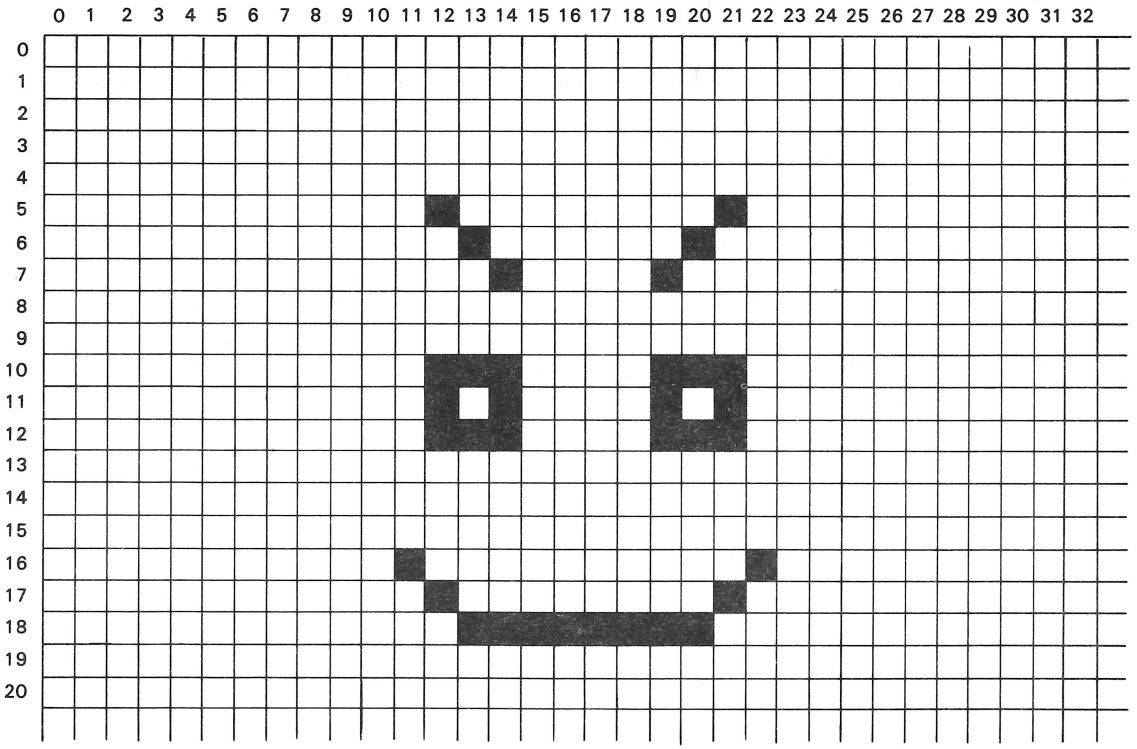


Challenge Programs

1. Write a program to draw a large plus sign near the center of the screen.

2. Make the two lines different colors. Even on a monochrome screen you should get some variation. *Hint: Add a second color line before the second line.*

3. Write a program that will draw the following design on the screen:



4. If you have a color monitor available, change the program you wrote for question three so that the eyebrows are brown, the eyes blue and the mouth red.

Area of a Rectangle in Graphics

The following program asks you to input the values for the length and width of a rectangle, graphs it in the color of your choice, then draws and calculates its area.

```
1 REM AREA OF A RECTANGLE
5 HOME
10 PRINT "THIS PROGRAM DRAWS A RECTANGLE AND CALCULATES
   ITS AREA."
20 PRINT "WHAT IS THE LENGTH (1-40) ";
30 INPUT L
40 PRINT "WHAT IS THE WIDTH (1-40) ";
50 INPUT W
60 IF L > 40 OR W > 40 THEN PRINT "CHOOSE VALUES 40 OR LESS.": GOTO
   20
70 PRINT "CHOOSE A COLOR (1-15)"
80 INPUT C
100 GR
110 COLOR= C
120 FOR Y = 0 TO L - 1
130 HLIN 0,W - 1 AT Y
140 NEXT Y
200 A = L*W
210 PRINT "THE AREA IS ";A;" SQUARE UNITS."
```

This program introduces the INPUT command. The INPUT command is used when the programmer wishes to have the user "input" some information. The computer will wait until the RETURN key has been pressed before actually storing the input. This response is then stored in a variable. A statement like INPUT A will accept the input, then store that information in a specific memory location. The contents of this variable can be changed every time the program is run (that is, a different response can be made to the same question).

One way to think of a variable is as a box where information is stored; when the program uses that variable, it goes to the box and looks at what is inside, then returns to the program with a report on what was in the box. In the above program, lines 30 and 50 ask the user to INPUT a length and a width, and line 80 asks him to INPUT a color. When the program is run, both of these values will be entered only after a RETURN has been pressed.

Challenge Problems

1. What would happen in line 60 if both L and W were 40 or greater?

2. Why are L –1 and W –1 used in lines 120 and 130, respectively?

Background Colors

The following program allows you to set the background color so you can draw your pictures with different screen colors. This procedure applies only to low resolution graphics.

```
10 REM BACKGROUND COLOR=YELLOW
20 GR : COLOR=13
30 FOR RW = 0 TO 39
40 HLIN 0,39 AT RW
50 NEXT RW
```

RW stands for row in the above example. In line 20 we set the computer to graphics mode and assign a color. In line 30 we tell the computer to start at row 0 and continue through row 39. Line 40 directs the computer to draw a horizontal line beginning at the first row (0). Line 50 tells the computer to continue looping through the program to draw on the second, third and fourth rows and so on until it reaches row 39.

Challenge Problems

1. What does REM stand for?

2. How does the computer respond when it sees the REM statement?

Challenge Programs

1. Try changing the background color to blue.

2. Put a time delay in the program so you can see each line being drawn on the screen.

3. Try to change the program so the computer colors the background with vertical lines instead of horizontal ones.

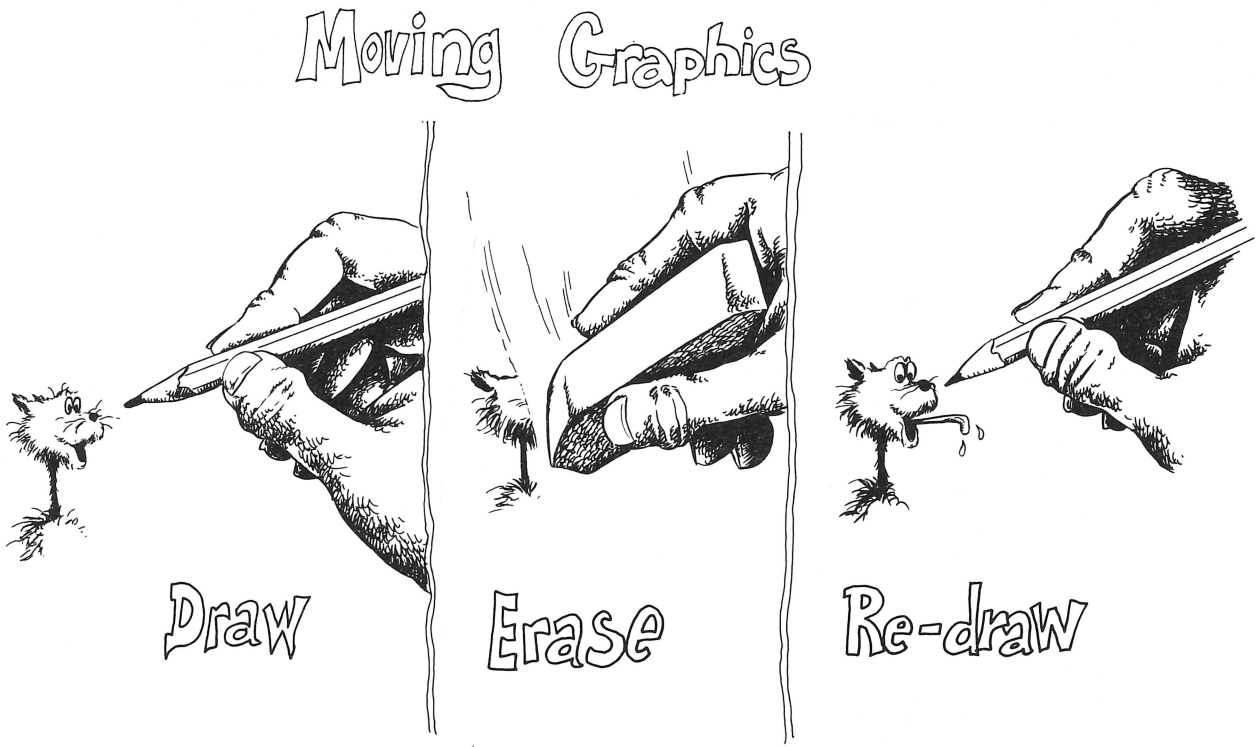
4. Make the top half of the screen blue and the bottom half green.

5. Put a cross or rectangle on top of the green area.

6. Change the program so it draws each line in a different color by looping through all the colors.

Moving Graphics

This program gives an object the appearance of moving across the screen. This is done by drawing an object, a cross in this case, and then erasing the cross. Then the cross is redrawn one square to the right. Since this gives the appearance of the same object at a different location the brain thinks that the object is moving.



Try the following program:

```
10 HOME : GR
20 FOR X = 1 TO 37
30 COLOR= 9
40 HLIN X,X + 2 AT 20
50 VLIN 19,21 AT X + 1
60 FOR W = 1 TO 200: NEXT W
70 COLOR=0
80 HLIN X,X + 2 AT 20
90 VLIN 19,21 AT X + 1
100 NEXT X
```

Lines 40 and 50 draw an orange cross. Line 70 sets the color to black and lines 80 and 90 draw the cross in black. This makes it appear as if the the cross is being erased because the background color is also black. Lines 20 and 100 move the cross one position to the right across the screen each time the loop is executed.

The following program draws a face that sticks its tongue in and out. The background is yellow.

```
5 REM MOVING TONGUE
10 HOME : GR : COLOR= 13
20 FOR RW = 0 TO 39
30 HLIN 0,39 AT RW
40 NEXT RW
100 COLOR= 2
110 HLIN 15,19 AT 16
115 HLIN 15,19 AT 20
120 VLIN 17,19 AT 15
125 VLIN 17,19 AT 19
130 PLOT 17,18
135 HLIN 21,25 AT 16
140 HLIN 21,25 AT 20
```

```
145 VLIN 17,19 AT 21
150 VLIN 17,19 AT 25
155 PLOT 23,18
160 PLOT 13,24: PLOT 14,25: PLOT 15,26
165 HLIN 16,24 AT 27
170 PLOT 25,26: PLOT 26,25: PLOT 27,24
200 HTAB 20: VTAB 22: PRINT "HI"
300 REM STICKS TONGUE OUT
310 COLOR= 1
315 VLIN 28,30 AT 20
320 FOR W = 1 TO 100: NEXT W
330 COLOR= 13: REM ERASES TONGUE
335 VLIN 28,30 AT 20
340 FOR W = 1 TO 100: NEXT W
350 C = C + 1: IF C < 50 GOTO 310
```

Challenge Problems

Refer to the previous program when working out the answers to the following questions.

1. What do lines 10 through 40 do?

2. What is the purpose of HTAB and VTAB in line 200?

3. What do lines 320 and 340 do?

4. What does line 350 do?

Challenge Programs

1. Change the background color to aquamarine and make the face magenta and the tongue yellow.

2. Make the tongue move faster, then slower.

3. Make the tongue stick out only 25 times.

4. Change the message at the bottom of the screen from "HI" to something else.

5. Graph an apple or a box with a worm popping in and out.

High Resolution Graphics

High resolution graphics allows you to draw pictures in great detail. In low resolution graphics the screen is divided into 40 squares by 40 squares. In high resolution graphics the screen is divided into a grid of 280 plotting points across by 160 plotting points down. These plotting points are called pixels.

A special command called HGR is used to set the screen for high resolution graphics. The commands for choosing a color and plotting points are similar to the commands used in lo-res graphics. Use HCOLOR= to choose a color and HPLOT to plot a point. For example, HCOLOR= 3 to select a color and and HPLOT 50,60 to plot a point. Use the word TO with the HPLOT command to draw lines, such as HPLOT 50,60 TO 120,100. This is very much like drawing dot to dot puzzles. You go from one point to the next point etc.

The following program draws a 5-pointed star:

```
10 HGR : REM 5-POINTED STAR
20 HCOLOR= 3
30 HPLOT 140,20 TO 175,129 TO 83,61 TO 197,61 TO 105,129 TO 140,20
```

Table of High Resolution Colors

0 Black 1	4 Black 2
1 Green	5 Orange
2 Violet	6 Blue
3 White 1	7 White 2

These colors may vary depending upon the TV or monitor used.

Challenge Problems

1. What command must be typed to return the screen to normal printing mode?

2. What does HGR stand for?

3. What does HCOLOR stand for?

4. Are HLIN and VLIN high resolution graphics commands?

Challenge Programs

1. Use high resolution graphics to write a program to draw a rectangle.

2. Write a program in high resolution graphics that draws a 6-pointed star.

3. Create a simple design of your own and write a program to draw it in hi-res graphics.

Note: A word of caution, however. If a line in your hi-res graphics program does not appear, and you have checked that you have not left out the HGR or the HCOLOR commands, try changing the HCOLOR. Due to the nature of TV tubes, some hi-res lines will not appear in some positions in certain colors. If you want to keep the color you have chosen, try moving the HPLOT points one point to the right.

Drawing in Hi-Res

Although hi-res graphics allows much more detailed drawing, it still requires point-to-point drawing. This means that curved lines or circular figures must be made up of a series of short straight line segments, giving it the appearance of a curve.

The following program draws a man in the moon in the middle of the screen:

```
10 HGR : HCOLOR= 3
20 HPLOT 155,55 TO 145,60 TO 130,65 TO 125,70 TO 120,80 TO 120,85
30 HPLOT TO 120,90 TO 125,100 TO 130,105 TO 135,110 TO 145,115
40 HPLOT TO 150,115 TO 155,115 TO 160,115 TO 165,110 TO 160,110
50 HPLOT TO 155,105 TO 150,105 TO 155,100 TO 150,90 TO 158,85 TO
   155,80
60 HPLOT TO 145,75 TO 145,70 TO 145,65 TO 150,60 TO 155,55
70 HPLOT 140,75 TO 138,75 TO 138,73 TO 140,73 TO 140,75
```

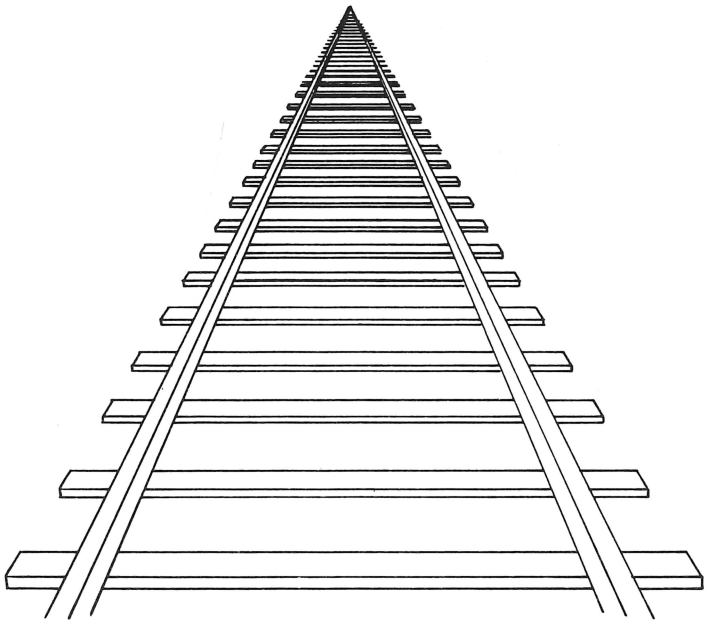
More Challenge Problems

1. Write a program to draw a house, tree and sun in the sky, using different colors for each object.

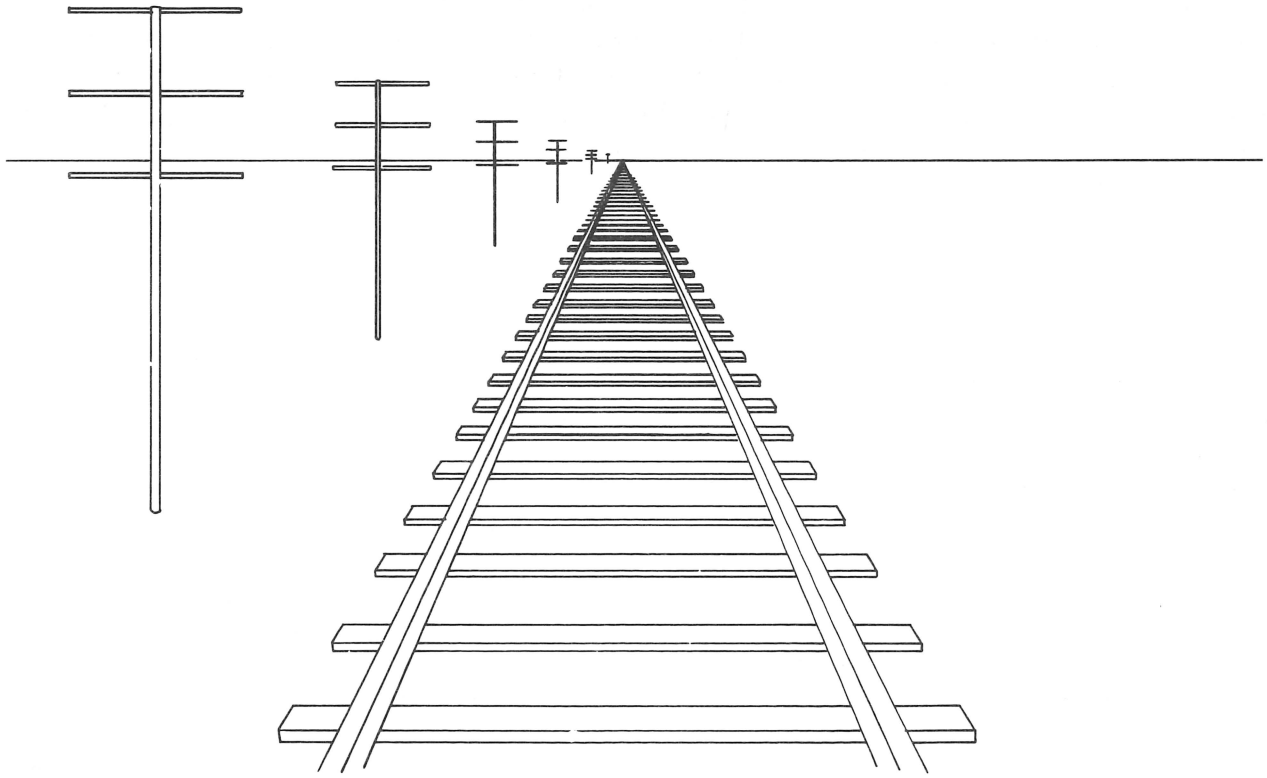
2. Create a scene of your own. Use time delays or subroutines to have parts of the scene appear at different times.

Three Dimensional Graphics

Drawing in 3-D requires a little forethought and planning. The basic idea is one of perspective; the farther away something is, the smaller it looks. Generally, a picture is drawn from a particular reference point and the farther an object is from the reference point, the smaller it should appear on the screen. All lines approach a vanishing point where the lines seem to converge. Look at the following sketch of a train track. The beginning of the track is the position at which the viewer is standing looking down the track. The farther you look down the track the smaller the track appears until it disappears into the distance (vanishing point).



You can also add a line for the horizon to give the picture the illusion of ground and sky.

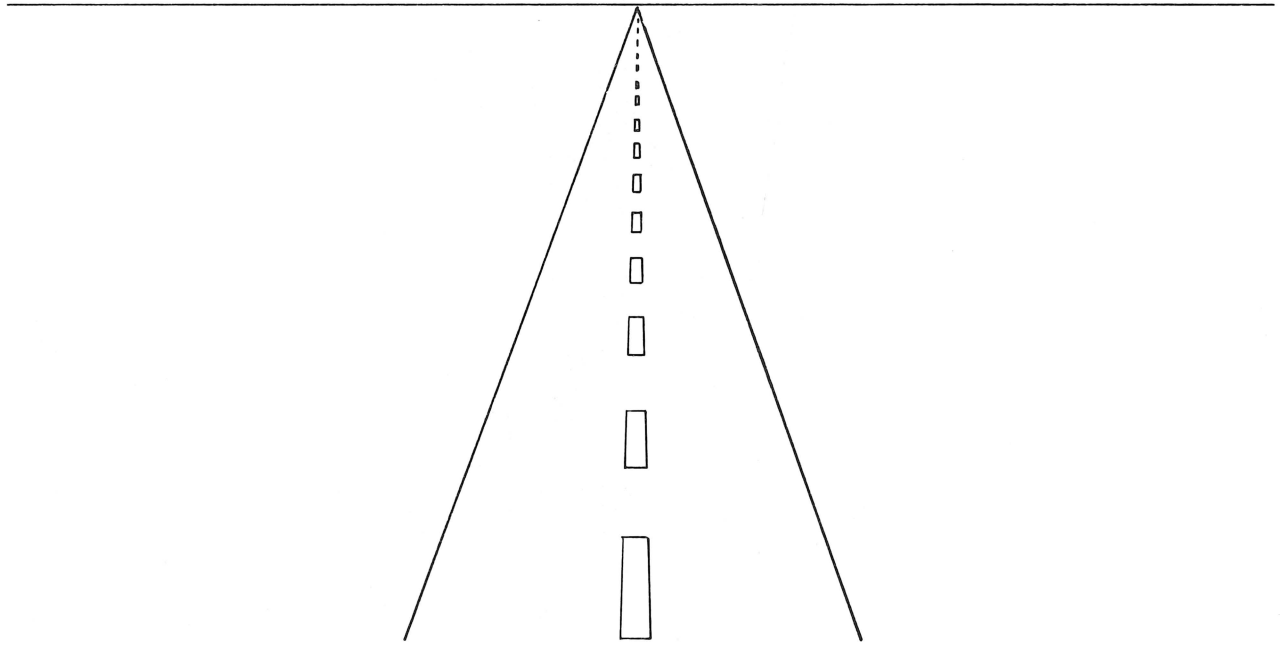


Vanishing Roads

The following program draws a road which appears to be three dimensional and vanishes at the horizon.

```
10 HGR
20 HCOLOR= 1
30 HPLOT 125,50 TO 50,155:HPLOT 125,50 TO 200,155:HPLOT 0,50 TO
   255,50
```

```
40 HPLOT 125,155 TO 125,130:HPLOT 125,120 TO 125,105:HPLOT
   125,98 TO 125,90:HPLOT 125,85 TO 125,79
50 HPLOT 125,74 TO 125,70:HPLOT 125,66 TO 125,62:HPLOT 125,59 TO
   125,57
60 HPLOT 125,55 TO 125,54
```

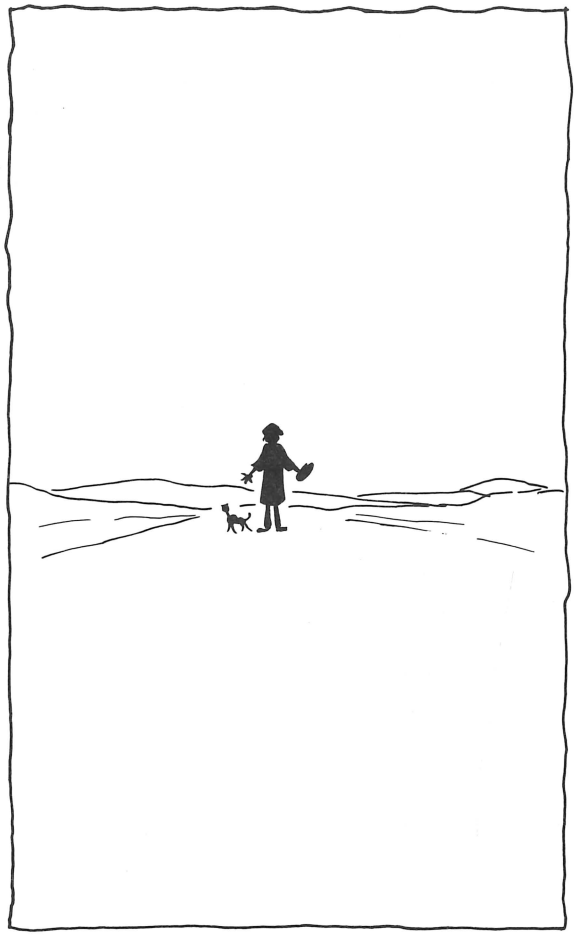


Note: If you have difficulty getting lines to appear in high resolution graphics, try changing the HCOLOR. Some colors don't show up as well as others, and some colors only excite odd numbered pixels while others excite even numbered pixels.

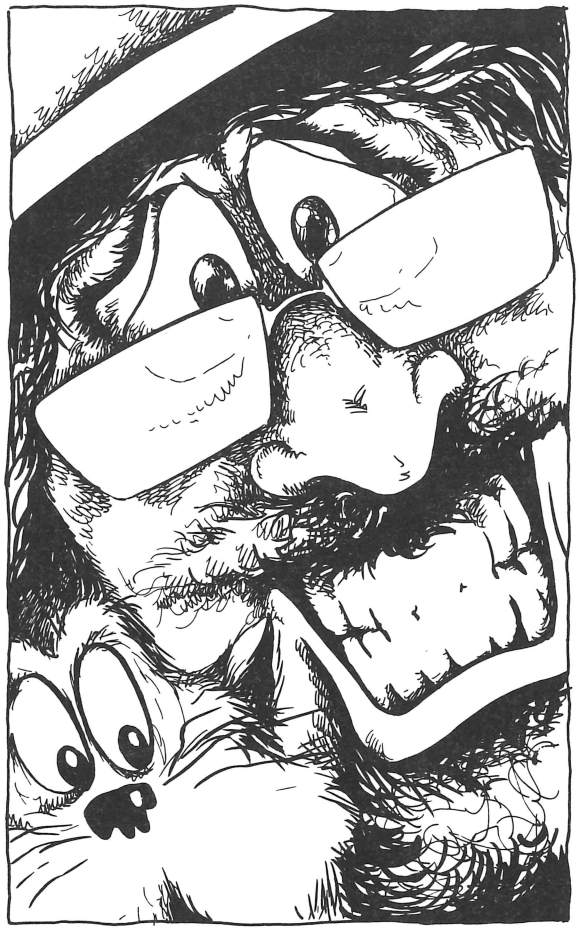
Now that you have a feeling for simple perspective drawing, you can build on this concept by adding other objects such as buildings, trees, cars, etc.

Just remember these points:

1. All lines meet at the vanishing points. Draw an X on some graph paper to divide the paper into four quardrants. You can use these lines to scale objects toward the vanishing point.



Far

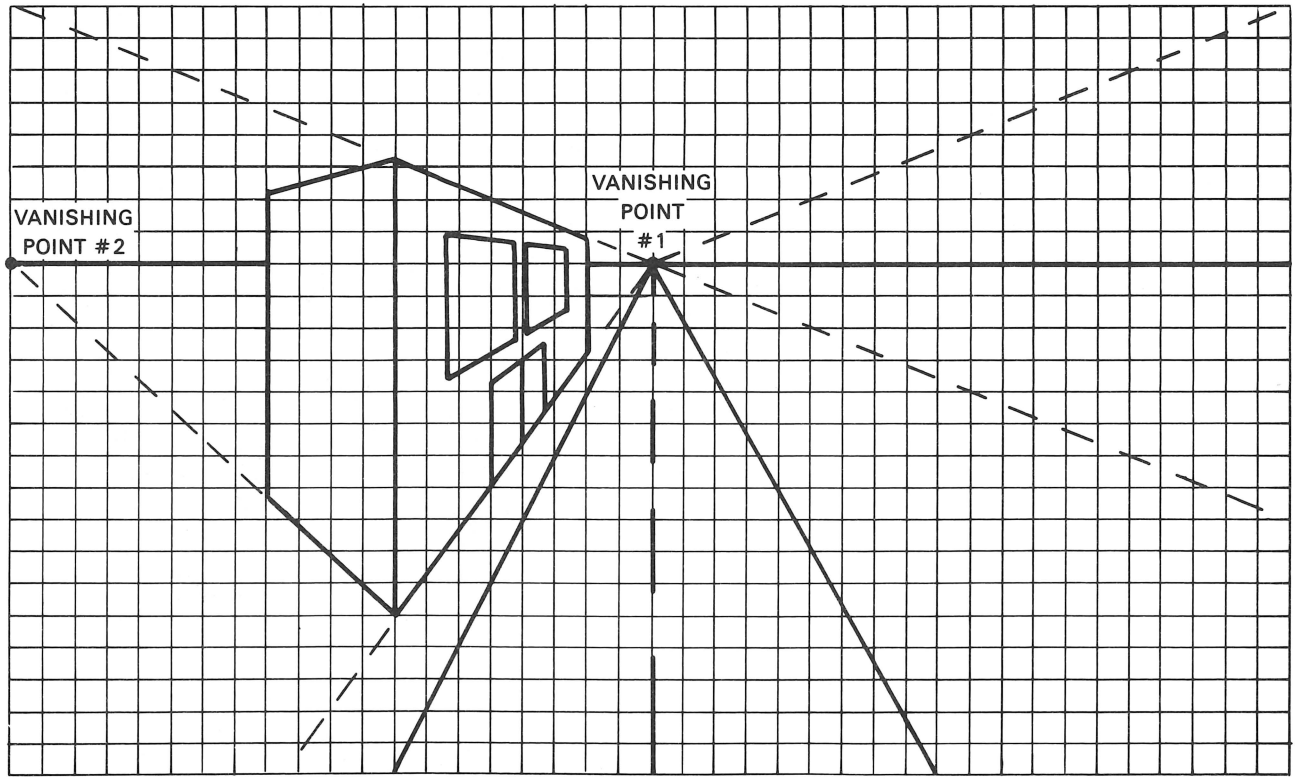


Near

2. The closer an object is to the viewer the larger it appears. Objects decrease in size as they approach the vanishing point.

Look at the following example. The graph was prepared with dotted lines that extend away from the vanishing point. Using these dotted lines, a road, horizon line, and building are added. The dotted lines are extended from the vanishing line to help get the perspective right. They are not a part of the picture.

A second vanishing point must be added to design the building. Using the two vanishing points you can determine the correct points for constructing the building.



In the following program perspective is used to give the illusion that the viewer is sitting in the cockpit of an airplane about to land on a runway. By drawing the building in such a way that the roof is visible to the viewer, an illusion of viewing the scene from above the ground is created.

```
5 HOME
10 HGR
20 HCOLOR= 1
25 REM RUNWAY
30 HPLOT 90,155 TO 145,5 TO 250,155
35 REM HANGER
40 HPLOT 121,40 TO 95,75 TO 65,50 TO 65,20 TO 95,30 TO 121,20 TO
   121,40
50 HPLOT 65,20 TO 95,15 TO 121,20 TO 121,40: HPLOT 95,75 TO 95,30
61 REM ARROWS
62 HPLOT 155,120 TO 155,80: HPLOT 147,85 TO 155,80 TO 164,85
63 REM ARROWS
64 HPLOT 151,60 TO 151,45: HPLOT 146,50 TO 151,45 TO 156,50
66 HPLOT 147,25 TO 147,19: HPLOT 144,21 TO 147,20 TO 150,21
74 REM RUNWAY LIGHTS
75 HCOLOR= 1
80 FOR X = 239 TO 150 STEP - 5
90 HPLOT X,X - 140
100 NEXT X
105 REM RUNWAY LIGHTS
110 FOR T = 1 TO 200: NEXT T
120 HCOLOR= 0
130 FOR X = 239 TO 150 STEP - 5
140 HPLOT X,X - 140
150 NEXT X
160 GOTO 75
```

An airline hanger is on the left. Lines 80 to 160 create plot points which appear to be runway lights. Notice that the section starting at line 75 with HCOLOR= 1 and the section starting at line 120 with HCOLOR= 0 are identical. The first section draws the lights, and the next section

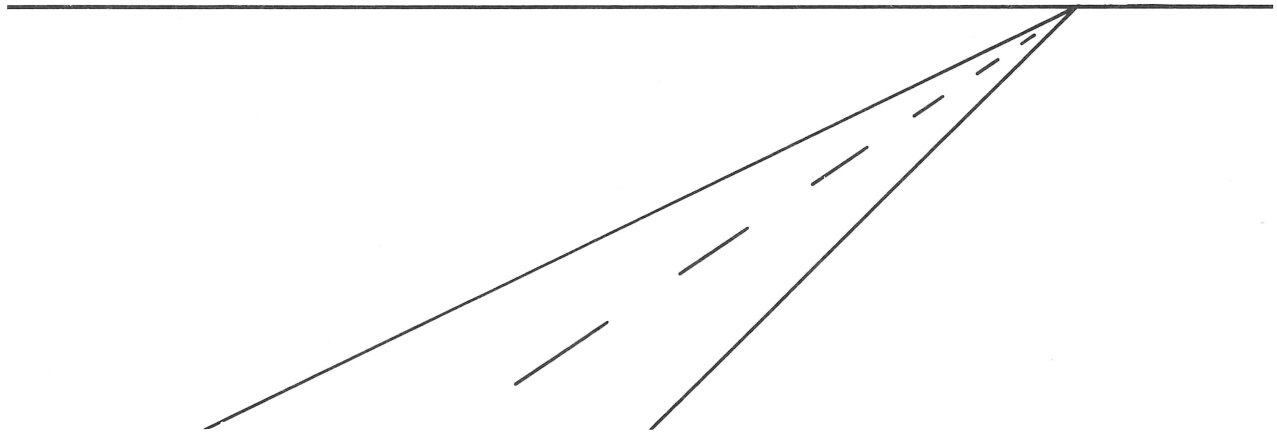
"erases" the lights. Actually the lights are redrawn in black which gives the impression that they have disappeared. In this case, drawing and "erasing" the lights gives the impression that the lights are blinking. The values in the FOR/NEXT loop are decreased because of the way the Applesoft coordinate system is set up. Graph numbers get smaller as they go from lower right toward the upper left approaching the vanishing point. At lines 90 and 140 the number 140 is subtracted from the second X coordinate to ensure that the points plotted are on the screen. This way you can avoid an illegal quantity error if the plotting point is out of range of the screen. The STEP command allows the programmer to plot every 5th point instead of every point, thus creating spaced dots rather than a line.

Challenge Problems

1. Add seagulls to the sky in the vanishing roads program.

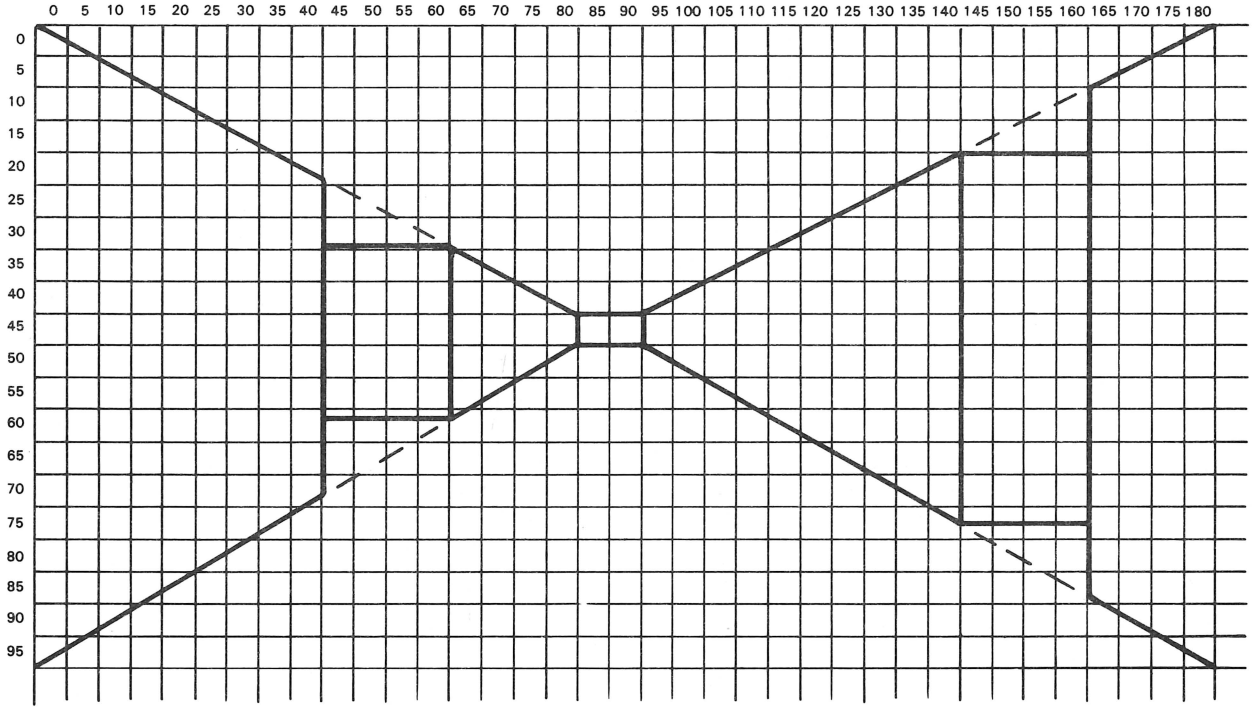
2. Add two objects, for example, a cactus, telephone pole, lizard, etc. Put one object near the viewer's standpoint, the other near the vanishing point. Remember, the object near the viewer has to be larger than the object near the vanishing point.

3. Change the direction of the road so that it starts on the left-hand side of the screen and vanishes near the upper right-hand corner. Note how this changes the viewer's position and perspective in the picture.



A-Mazing Graphics

Designing a three dimensional display for a maze also requires the use of perspective. Use an X to divide a sheet of graph paper into four quadrants. The lines that converge at the vanishing point can be used to construct the maze, as shown in the following example.

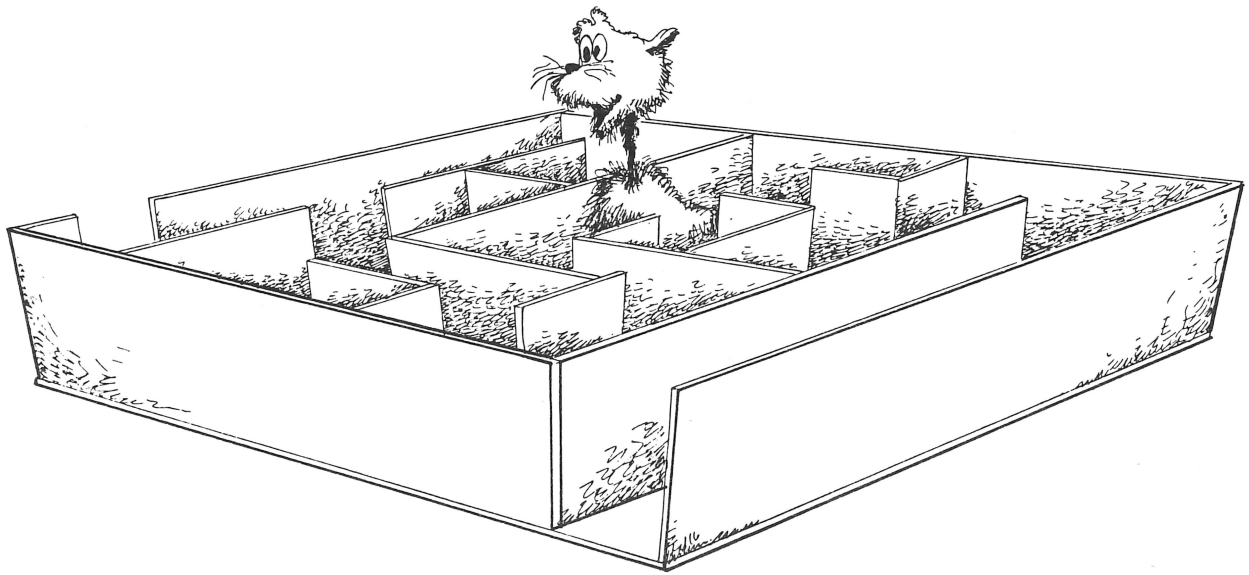


The dotted line is the X we used to divide the graph into 4 quadrants in order to establish depth (vanishing point).

Maze Magic

The following program draws a maze. The perspective makes it seem as though you are standing on the right side of the maze looking toward the end of the hallway.

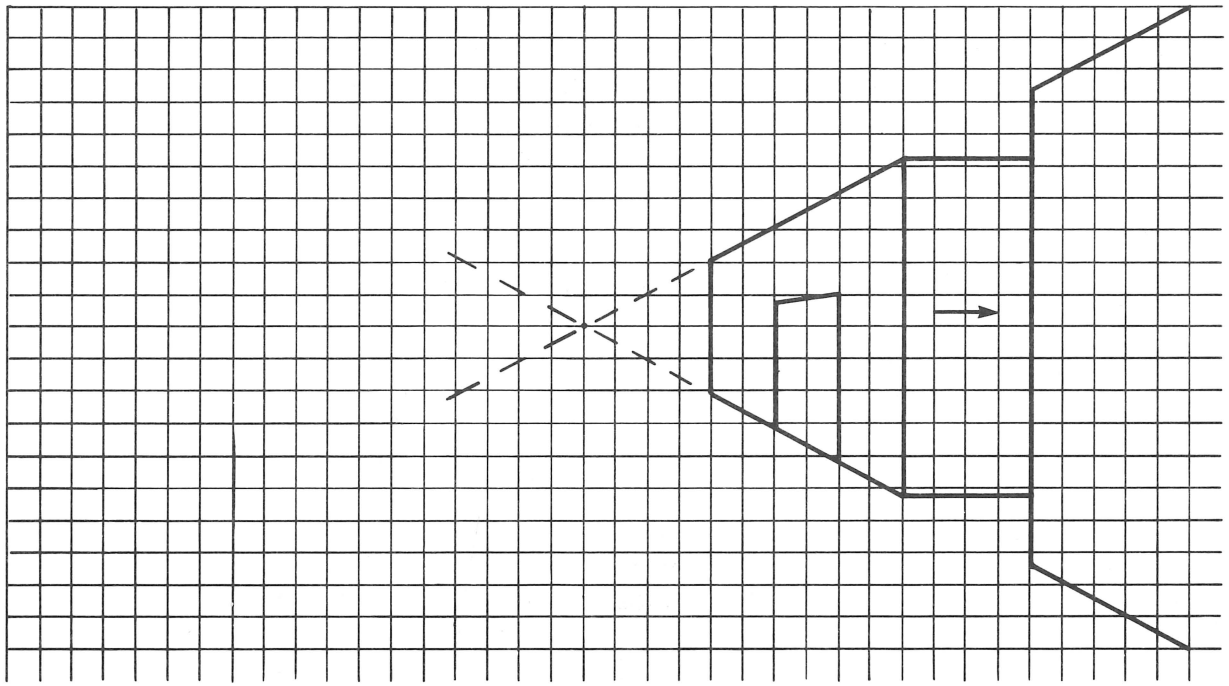
```
1 HOME
10 HGR
20 COLOR= 1
30 HPLOT 0,0 TO 45,25 TO 45,75 TO 0,100
40 HPLOT 45,35 TO 65,35 TO 65,65 TO 45,65
50 HPLOT 65,35 TO 85,45 TO 85,50 TO 65,65
60 HPLOT 85,48 TO 95,48 TO 95,45 TO 85,45
70 HPLOT 95,45 TO 145,15 TO 145,80 TO 95,48
80 HPLOT 145,15 TO 165,15 TO 165,80 TO 145,80
90 HPLOT 180,0 TO 165,5 TO 165,90 TO 220,125
```



Challenge Problems

1. Construct a maze of your own. Add a corridor on the right-hand side.

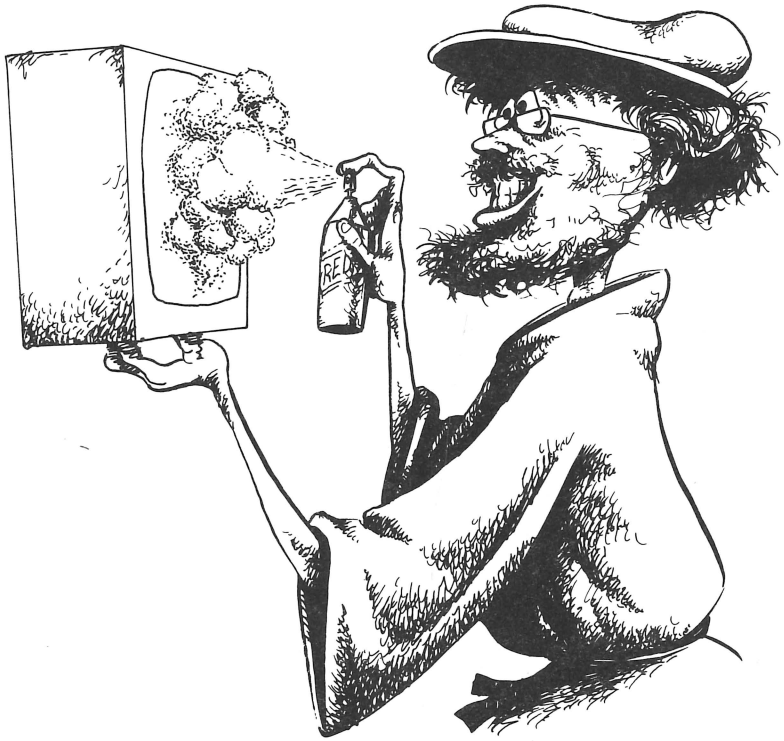
2. Put an arrow on one of the corridor's walls as a directional indicator as follows:



3. Add two or more doors at different points in the maze. Remember to use the vanishing point to draw the doors — the door will be a trapezoid instead of a rectangle in perspective drawing.

Hi-Res Background Colors

In order to change the background or screen color in high resolution graphics you need to set the computer for hi-res graphics, set the color you want for the background, HPLOT any point on the screen, and then call a particular memory location.



Changing Screen Color!

The following program sets the background color to green:

```
10 HGR
20 HCOLOR= 1
30 HPLOT 80,80
40 CALL -3082
```

There are two screens in high resolution graphics. Screen one is accessed by HGR and consists of a 280 x 160 grid screen, with the bottom 4 lines reserved for text. The second graphics screen is accessed by HGR2. The HGR2 screen allows use of the entire screen, and reserves no space for text. It consists of a 280 x 192 grid.

Challenge Problems

1. How many high resolution screens are there?

2. How many HCOLORs are there?

3. What are the coordinates for the center of an HGR screen?

4. What are the coordinates for the center of an HGR2 screen?

Challenge Programs

1. Change the background color to violet.

2. Write a program that will color the second graphics screen purple.

3. Have the screen loop through all the hi-res colors.

4. Draw a box, or other shape, on a blue background.

Hi-Res Action

To create action in hi-res graphics using only graphics screen one, you can create a design, place it in a subroutine, then change the color from a color to black and then back again.

The following program draws a box with a worm popping out.

```
5 HOME : HGR
10 HCOLOR= 6
20 HPLOT 120,60 TO 120,100 TO 160,100 TO 160,60 TO 120,60
30 HCOLOR= 3
40 GOSUB 110
50 HCOLOR= 0
60 GOSUB 110
70 C = C + 1
80 IF C = 10 THEN TEXT : HOME : END
90 GOTO 30
110 HPLOT 135,59 TO 140,55 TO 135,50 TO 140,45 TO 135,40 TO 140,
    35 TO 150,35
120 HPLOT TO 155,40 TO 150,45 TO 155,50 TO 150,55 TO 155,59
130 HPLOT 143,40: HPLOT 148,40
140 FOR T = 1 TO 300: NEXT T
150 RETURN
```

Challenge Problems

1. What does line 70 do?

2. What is the purpose of line 50?

3. Why does line 120 begin with HPLOT TO?

Challenge Programs

1. Change the speed at which the worm pops out of the box. Make it go slower and then faster.

2. Change the program so the worm comes out 20 times.

3. Change the program so that the worm changes color instead of popping out.

4. Design a different figure to pop out of the box.

Alternating Graphics Pages One and Two

The Apple computer has two memory blocks (called pages) that are used for hi-res graphics. By switching between graphics pages one and two, you can set up sophisticated movement routines. This is done in a way very much like the way cartoons are made. Each successive picture in the series has a small difference from the previous picture. If the pictures are flipped through rapidly an illusion of movement is created. Little cartoon flip books are excellent examples of this process.

The following program shows a simplified use of page one and two graphics:

```
10 HGR : HCOLOR= 3
20 HPLOT 130,70 TO 150,70 TO 150,80 TO 140,90 TO 130,80 TO 130,70
30 HPLOT 138,75: HPLOT 142,75
40 HPLOT 150,80 TO 160,70 TO 170,75
50 HPLOT 130,80 TO 120,70 TO 110,75
60 HGR2 : HCOLOR= 3
70 HPLOT 130,70 TO 150,70 TO 150,80 TO 140,90 TO 130,80 TO 130,70
80 HPLOT 138,75: HPLOT 142,75
90 HPLOT 150,80 TO 160,85 TO 170,90
100 HPLOT 130,80 TO 120,85 TO 110,90
110 C = C + 1
120 POKE - 16300,0
130 FOR T = 1 TO 200: NEXT T
140 POKE - 16299,0
150 FOR T = 1 TO 200: NEXT T
160 IF C = 20 THEN TEXT : HOME : GOTO 200
170 GOTO 110
200 VTAB 12
```

```
210 PRINT "**********************";
220 FLASH : PRINT "BYE";: NORMAL
230 PRINT "**********************"
```

Challenge Problems

1. How many times does the bird flap before BYE is displayed?

2. Which lines draw the picture for page 2?

Note: POKE -16300,0 switches the screen from graphics page 2 (HGR2) to graphics page 1 (HGR). POKE -16299,0 switches the screen from page 1 to 2.

Moving Graphics — Hungry Man

Here is another example of using the two graphics pages to obtain the appearance of movement.

```
10 HGR : POKE - 16302,0: HGR2
30 POKE 230,32
40 HCOLOR= 3
50 GOSUB 1110
60 HCOLOR= 0
70 GOSUB 1210
80 HCOLOR= 3
90 GOSUB 1310
100 FOR W = 1 TO 250: NEXT W
110 POKE - 16300,0
120 POKE 230,64
130 HCOLOR= 3
140 GOSUB 1110
150 HCOLOR= 0
160 GOSUB 1310
170 HCOLOR= 3
180 GOSUB 1210
```

```
190 FOR W = 1 TO 250: NEXT W
200 POKE - 16299,0
210 GOTO 30
1110 HPLOT 190,95 TO 190,80 TO 200,80 TO 190,60 TO 190,40 TO 170,40
1120 HPLOT 190,40 TO 190,10 TO 90,10 TO 90,158 TO 160,158 TO 160,150
     TO 190,150 TO 190,135
1130 HPLOT 130,55 TO 120,55 TO 120,65 TO 126,65 TO 126,70
1140 RETURN
1210 HPLOT 190,95 TO 150,115 TO 190,135
1220 HPLOT 174,110 TO 184,110 TO 184,120 TO 174,120 TO 174,110
1230 RETURN
1310 HPLOT 190,95 TO 190,135
1320 HPLOT 190,115 TO 150,115
1330 HPLOT 200,110 TO 210,110 TO 210,120 TO 200,120 TO 200,110
1340 RETURN
```

In this program, the picture is stored as a series of subroutines. By drawing and "undrawing" the picture, parts of the picture appear to move.

POKE – 16302,0 sets the entire screen for graphics, leaving no space at the bottom for text. POKE 230,32 allows you to draw on page one and POKE 230,64 allows you to draw on page two. This is not necessary for a static movement program, but will be useful if you want figures to change position on the screen. This eliminates possible flickering when a picture is run.

More Challenge Problems

1. What word is used at the end of a subroutine to continue the program?
2. How many lines are left at the bottom of the screen for text in HGR?
3. How many lines are left at the bottom of the screen for text in HGR2?
4. What is the function of POKE -16300,0 and POKE -16299,0?
5. What function do lines 60 and 150 perform?

Challenge Programs

1. Make the hungry man eat a long rectangle instead of a square.
2. Take out the POKE which removes the text lines, and add a message to the bottom of the picture.
3. Design your own face with an eye that winks.

Hint: Lines 1110 to 1130 draw the face, ear, eye, and nose. Lines 1210 to 1230 draw the open mouth, and 1310 to 1320 draw the closed mouth.

Drawing in Hi-Res with Paddles

You can set up your program to read paddle or joystick positions, and draw hi-res lines according to the settings. The following program uses both paddles (PDL (0) and PDL (1)) or a joystick.

```
5 HOME : PRINT "WHENEVER YOU WANT TO CLEAR THE SCREEN AFTER
  DRAWING, PRESS ANY KEY ON THE KEYBOARD."
10 FOR T = 1 TO 8000: NEXT T
20 HGR : POKE - 16302,0
30 X = 140:Y = 96
40 HCOLOR=3
50 HPLOT X,Y
60 X = PDL (0) * 279 / 255
70 Y = PDL (1) * 191 / 255
80 IF X > 0 AND X < 279 AND Y > 0 AND Y < 191 THEN 100
90 GOTO 50
100 HPLOT TO X,Y
110 Z = PEEK ( - 16384)
120 IF Z > 127 THEN POKE - 16368,0: GOTO 20
130 GOTO 60
```

Line 30 sets a starting point for the first dot. Line 50 tells the computer to plot those points. Lines 60 and 70 read the paddle positions and select new positions for screen drawing. The PEEK in line 110 checks to see if any key on the keyboard has been pressed. If it has, then POKE –16368,0 in line 120 resets the keyboard and GOTO 20 starts the program over. Otherwise GOTO 60 continues to draw the line.

Challenge Problems

1. What does POKE –16302,0 do?

2. Add something to this program that lets you end or get out of the program without hitting reset.

3. Reset the screen size so that it only draws on a 100 x 100 grid in the center of the screen.

Using Joystick/Paddle to Fire Missiles

You can use the paddle button to fire missiles and the knob to move across the bottom of the screen. The following program sends a "bullet" up the screen. This technique may be used for firing at targets in large games.

```
10 HOME : HGR
12 VTAB 23
15 PRINT "USE PADDLE 0"
20 X = PDL (0) * 279 / 255
25 IF PEEK ( - 16287) > 127 THEN GOSUB 35
30 GOTO 20
35 FOR I = 159 TO 1 STEP - 1
40 HCOLOR= 3
45 HPLOT X,I
60 HCOLOR= 0: HPLOT X,I
65 NEXT I
70 RETURN
```

Make the following changes in the above program to get a laser beam effect:

Change line 40 to line 32 (be sure to delete line 40)
Add a new line: 50 NEXT I
Add a new line: 55 FOR I = 159 TO 1 STEP - 1
Change the GOSUB 35 in line 25 to GOSUB 32

The first number of the fraction 279/255 defines the range of the screen. Since HGR contains 280 horizontal plotting points, the values which can be used are 0-279.

PEEK (– 16287) has the computer check to see if the paddle button has been pressed. If the value is greater than 127, the computer knows the button was pressed, and then goes to the subroutine to fire the missile.

Challenge Problems

1. What does line 35 do?

2. Why do you HPLOT X,I twice, once at line 45 and again at 60?

3. What would you have to change 279/255 to in order to write the same program in lo-res graphics?

Challenge Programs

1. Make the program fire the missiles from the top to the bottom of the screen.

2. Change the program to run in low resolution graphics.

Shooting Random Targets

We have chosen low resolution graphics to demostrate the principles of creating random targets and having paddle controlled missiles shoot at them. Using lo-res graphics makes it easier to manipulate and to see the action without lengthy drawing of figures or designs.

The following program demonstrates setting random plots on the top half of the screen, with a missile firing from the bottom.

```
10 HOME : GR
15 VTAB 22
20 PRINT "USE PADDLE 0"
30 N = INT ( RND (1) * 40)
40 Y = INT ( RND (1) * 15)
50 X = PDL (0) * 39 / 255
60 X = INT (X)
70 IF PEEK ( - 16287) > 127 THEN GOSUB 200
80 COLOR= 0: PLOT N,Y
90 GOTO 30
200 FOR I = 39 TO 1 STEP - 1
210 COLOR= 10: PLOT N,Y
220 COLOR= 8: PLOT X,I
230 IF N = X THEN IF Y = I THEN PRINT CHR$ (7):C = C + 1
240 COLOR= 0: PLOT X,I
250 VTAB 22: HTAB 20: PRINT "NUMBER OF HITS  ";C
260 NEXT I
270 RETURN
```

Lines 30 and 40 use the random number generator to ask the computer to select two random numbers and store them in variables N and Y. Line 50 reads the paddle's position, then line 60 converts that value to an integer. Line 230 has the computer check to see if the missile occupies the same space as the random plot. If it does, it beeps, CHR$(7), then keeps count, C = C + 1, of the hits.

For a guided missile effect, change lines 50 and 60 into lines 212 and 215. Use the paddle or joystick to guide the movement of the missile. Remember to delete the old lines 50 and 60.

Challenge Problems

1. What does line 70 do?

2. What do lines 80 and 240 do?

3. What is CHR$(7)?

Challenge Programs

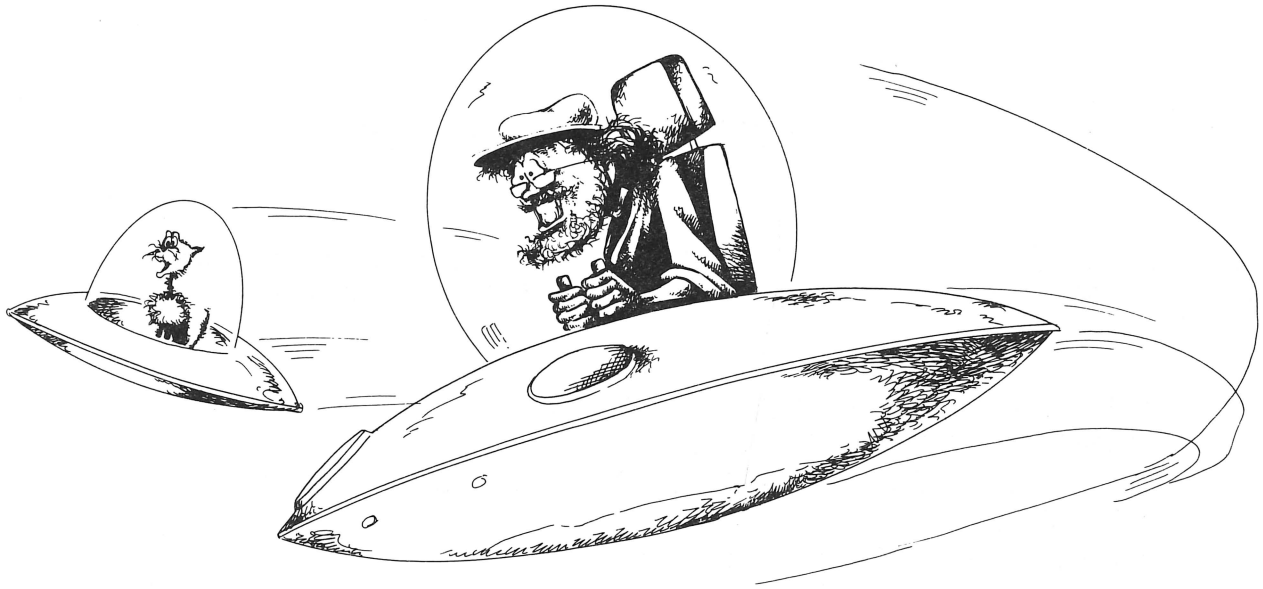
1. Change the program so that the random dots come down further down on the screen.

2. Change the horizontal screen range so that the dots appear in a narrower column of the screen so that it is easier to play.

U.F.O.

This program draws a flying saucer complete with blinking lights. The use of multiple time delays after HPLOTting each saucer light makes the lights appear to blink.

```
10 HGR : HCOLOR= 1
20 HPLOT 150,90 TO 125,75 TO 100,90
30 HPLOT 110,100 TO 140,100
35 HPLOT 150,90 TO 140,100: HPLOT 100,90 TO 110,100
50 HCOLOR= 1
51 FOR X = 100 TO 145 STEP 5:Y = 90
52 HPLOT X,Y TO X + 5,Y: NEXT X
60 HPLOT 135,95
61 FOR T = 1 TO 50: NEXT T
62 HPLOT 131,95
63 FOR T = 1 TO 50: NEXT T
64 HPLOT 125,95
65 FOR T = 1 TO 50: NEXT T
66 HPLOT 121,95
67 FOR T = 1 TO 50: NEXT T
68 HPLOT 115,95
69 FOR T = 1 TO 50: NEXT T
70 HPLOT 111,95
71 FOR T = 1 TO 200: NEXT T
72 HCOLOR= 0
73 FOR X = 100 TO 150:Y = 90: HPLOT X,Y: NEXT X
80 HPLOT 135,95: HPLOT 131,95: HPLOT 125,95: HPLOT 121,95: HPLOT
   115,95: HPLOT 111,95
85 FOR T = 1 TO 50: NEXT T
90 GOTO 50
```



Lines 51 and 52 draw the center line across the saucer. The line is drawn from left to right. Lines 72 and 73 "undraw" the center line. Using the FOR/NEXT loop can create the illusion of movement. In this case the ship looks like it is spinning.

Close Encounters

A simple way of making an object grow or appear to move toward you, is assigning a variable to each beginning plotting point. Then by using a FOR/NEXT loop, you can increase or decrease each coordinate. This makes the object appear to be closer since it has increased in size. In the following program the flying saucer from the U.F.O. program is used, but now it has the appearance of approaching the viewer.

```
10 HGR
15 A = 150:B = 90:C = 125:D = 75:E = 100: F = 90
17 L = 150: M= 90:N = 140:O = 100:P = 110:Q = 100:R = 100:S = 90
18 FOR X = 0 TO 50 STEP 5
20 HCOLOR= 1
25 HPLOT A + X,B TO C,D - X TO E - X,F
30 HPLOT L + X,M TO N + X,O TO P - X,Q TO R - X,S
40 HPLOT E - X,F TO L + X,M
65 HCOLOR= 0
70 HPLOT A + X,B TO C,D - X TO E - X,F
80 HPLOT L + X,M TO N + X,O TO P - X,Q TO R - X,S
90 HPLOT E - X,F TO L + X,M
110 NEXT X
120 HCOLOR=1
130 HPLOT A + X,B TO C,D - X TO E - X,F
140 HPLOT L + X,M TO N + X,O TO P - X,Q TO R - X,S
150 HPLOT E - X,F TO L + X,M
```

The method of adding or subtracting units of 5 is simply a matter of estimating the direction (+ or –) on the graph paper, and moving all points the same value (X) so that all parts of the drawing increase equally. If you are unfamiliar with graphing, you will notice that if you draw a ver-

tical line through the center of your object, and a horizontal line through the center of the object you create four boxes or quadrants. If you enlarge a part of the drawing in the upper right quadrant, then the X (horizontal) number will increase by the amount selected, and the Y (vertical) number will decrease by the same amount. To accomplish this task of using variables and increasing them to create an image enlargement, it is important that you understand how the numbers increase or decrease as you change positions on the graph. If this concept is not clear, get out a piece of graph paper and draw two squares. Make one of the squares small and put it inside the other larger square. Plot the corner points of both squares and see how the numbers from each corner of the inner square relate to the corner of the corresponding outer square.

Sound Effects for Games

The computer can produce a variety of sound effects which can be added to your program. The following program allows you to try out different sounds by INPUTing different pitch and length values.

```
5 HOME
10 FOR X = 848 TO 870: READ Y
15 POKE X,Y: NEXT X
20 INPUT "PITCH= ";P
25 INPUT "LENGTH= ";L
30 POKE 849,L: POKE 855,P
35 FOR N = 1 TO 10: CALL 848
40 NEXT N
45 GOTO 20
100 DATA 160,32,152,32,95,3,73,255,32,95,3,136,208,244,96,170,202,
    208,253,44,48,192,96
```

Caution: POKE commands force information into specific memory locations. Be sure to check the numbers in your POKE commands before running the program, since a number POKEd into a wrong location could cause problems with information already stored there. If you do this and the program still does not run, double check the values in the DATA statements.

Lines 20 and 25 allow you to put numbers in from the keyboard for the pitch and length. These values must be between 0 and 255. Line 30 POKEs into memory the length and the pitch. The first part of line 35 (FOR N = 1 TO 10) sets up a warble in the sound produced. CALL 848 tells the computer to sound the speaker.

PHILIP GLASS

Once you have tested the sounds and have selected a sound you like, you may alter the program so it produces the desired sound. To do this delete lines 20, 25, and 45. Then in line 30 substitute the length number for L and the pitch number for P.

If you want a sound that doesn't warble, you can also delete the FOR/NEXT commands in lines 35 and 40, but keep the CALL 848.

Lines 10, 15 and 100 are always required for sound as they set up a routine that allows the computer to execute sound.

Challenge Programs

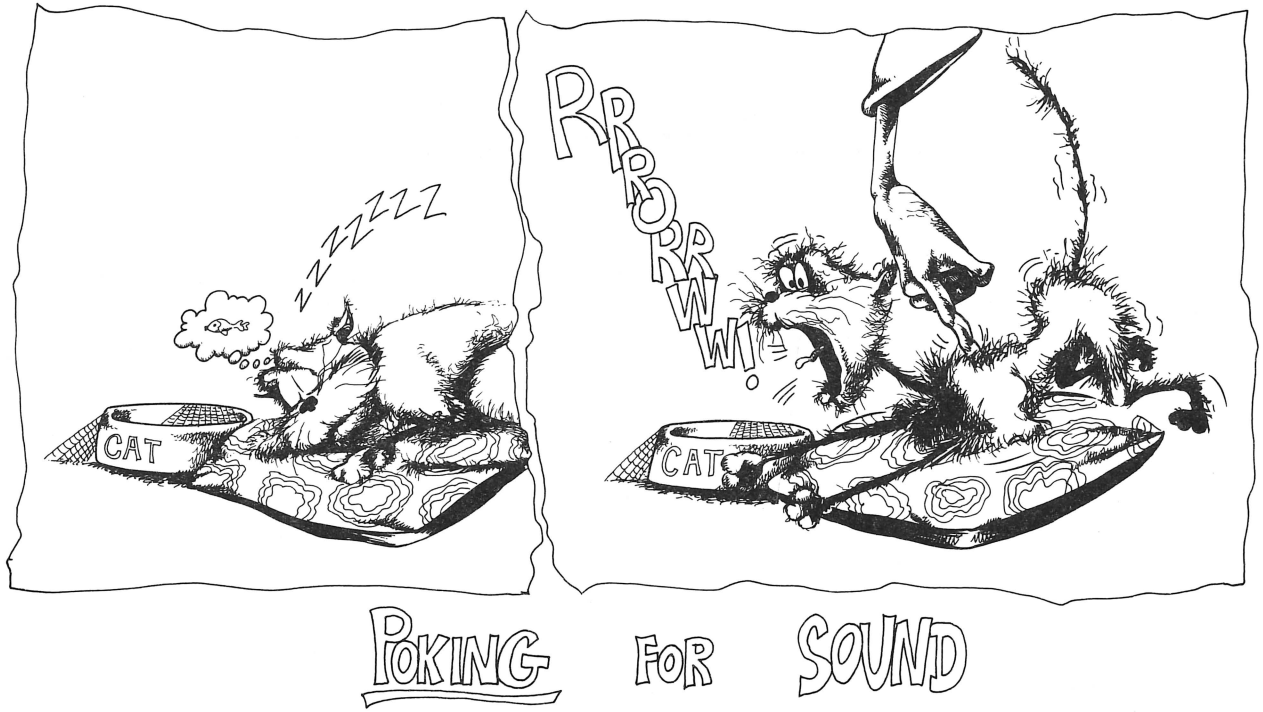
1. Select a sound, and write a program that produces only that one sound and then ends.

2. Change the program so it makes an even sound without the warble.

Sound Subroutines

If you want to include sound effects in your program, put the basic sound routine in a subroutine and POKE it in once at the beginning of your program. This will keep the speaker routine in memory, and allow you to write smaller subroutines for different sounds. A short example is as follows:

```
5 HOME : GOSUB 500
10 PRINT "WHAT DOES CRT STAND FOR?"
20 INPUT A$
30 IF A$ = "CATHODE RAY TUBE" THEN FLASH : PRINT "CORRECT":
   NORMAL : GOSUB 1000
40 IF A$ <> "CATHODE RAY TUBE" THEN PRINT "WRONG": GOSUB 400
50 PRINT "WHAT DOES RAM STAND FOR?"
60 INPUT B$
70 IF B$ = "RANDOM ACCESS MEMORY" THEN INVERSE : PRINT "RIGHT!":
   NORMAL : GOSUB 1000
80 IF B$ <> "RANDOM ACCESS MEMORY" THEN PRINT "SORRY": GOSUB
   400
399 END
400 POKE 849,244: POKE 855,244
410 CALL 848
420 RETURN
500 FOR X = 848 TO 870: READ Y
510 POKE X,Y: NEXT X
520 DATA 160,32,152,32,95,3,73,255,32,95,3,136,208,244,96,170,202,
    208,253,44,48,192,96
530 RETURN
1000 PRINT CHR$(7); CHR$(7); CHR$ (7)
1100 RETURN
```



The subroutine at line 500 does not make sound in itself, but it is required to set up the speaker to produce sound. The actual sound is set at line 400. You can POKE 849,X (X = any number between 0-255 for length), and POKE 855,X (X = any number between 0 and 255 for pitch). Then you must CALL 848 to hear the sound. The body of the program should end before reaching the subroutines so that the computer does not continue through subroutines after finishing the questions. This is done by placing an END at line 399.

Challenge Problems

1. How many times will the bell ring if you get the right answer?

2. What appears at the end of each subroutine in this program?

3. How many subroutines are in this program?

Challenge Programs

1. Add eight more questions about computers to this program.

2. Use IF/THEN statements to evaluate the answers.

3. Write one additional subroutine using graphics or sound and access it at least twice during the program.

Introduction to Apple Music

We must use a subroutine to prepare the speaker for sound. For music the subroutine is as follows:

```
FOR A = 771 TO 789 : READ D
POKE A,D : NEXT A
DATA 173,48,192,136,208,4
DATA 198,1,240,8,202,208
DATA 246,166,0,76,3,3,96
```

As we said before, this routine only needs to be POKEd into memory once at the beginning of the program. You may combine the DATA numbers into one line, but they must appear in the same order. Remember this routine does not produce sound, it only prepares the speaker for sound. Next you must select pitch and length values to make the desired sounds.

The following chart lists the musical notes and their numeric values.

Pitch Table

Note	Pitch Value
B	202
C	191
C# or D♭	180
D	170
D# or E♭	161
E	152
F	143
F# or G♭	135
G	128
G# or A♭	120
A	114
A# or B♭	107
B	101
C	96
C# or D♭	90
D	85
D# or E♭	80
E	76

To make a simple melody, put a series of notes and their length values in DATA statements. The length value of an eighth note is approximately 48. The value of a quarter note is between 80-90. Length values are not exact multiples of each other, so if an eighth note is 48, it does not necessarily follow that a half note would be 4 x 48 or 192. You should experiment with length values to determine what sounds best.

Charge into Music

The following program plays the "charge" tune:

```
10 FOR A = 771 TO 789: READ D
20 POKE A,D: NEXT A
30 DATA 173,48,192,136,208,4,198,1,240,8,202,208,246,166,0,76,3,3,96
50 FOR S = 1 TO 6
60 READ P,L
70 POKE 0,P: POKE 1,L: CALL 771
80 NEXT S
100 DATA 143,48,128,48,114,48,96,90,114,48,96,250
```

Lines 10 through 30 set the speaker for music. Line 50 sets the number of notes in the tune. Line 60 reads the pitch and length values from the DATA statement (the values are entered in the DATA statement in the form pitch, length). Therefore the first note of 143,48 means the pitch is 143 (F), and the length is 48 (an eighth note). The next two numbers — 128,48 — are the pitch and length of the second note.

Challenge Problems

1. If there were 12 notes in the song, what would line 50 be?

2. What does CALL 771 do?

Challenge Programs

1. Write a short tune with six to ten notes in it. (You may want to find the notes on a piano or xylophone.)

2. Write a program that will play your tune on the computer.

Space Music

The following program is another example of Apple music.

```
10 REM * STAR WARS THEME *
20 FOR A = 771 TO 789: READ D
30 POKE A,D: NEXT A
40 DATA 173,48,192,136,208,4,198,1,240,8,202,208,246,166,0,76,3,3,96
100 FOR S = 1 TO 19
110 READ P,L
120 POKE 0,P: POKE 1,L: CALL 771
130 NEXT S
140 DATA 107,48,107,48,107,48
150 DATA 80,240,53,240
160 DATA 60,48,64,48,71,48
170 DATA 40,240,53,240
180 DATA 60,48,64,48,71,48
190 DATA 40,240,53,240
200 DATA 60,48,64,48,60,48
210 DATA 71,255
220 RESTORE
230 C = C + 1
240 IF C = 2 THEN END
250 GOTO 20
```

Challenge Problems

1. What is the purpose of the routine that is set up in lines 20-40?

2. In line 100 there is a statement, FOR S = 1 TO 19. To what does the 1 TO 19 refer?

3. What does the first number in line 140 stand for?

4. What does the second number in line 140 stand for?

5. What does the word RESTORE do?

6. For what purpose is a counter statement used in this program?

Challenge Programs

1. Write an adventure or mystery game program. Include a short tune at the beginning of the program. Make sure your program begins with a title page. Store the tune in a subroutine so you can play it once during the program or at the end.

Combining Graphics and Sound

The following program combines graphics with a sound program:

```
5 FOR N = 771 TO 789: READ R
6 POKE N,R: NEXT N
8 DATA 173,48,192,136,208,4,198,1,240,8,202,208,246,166,0,76,3,3,96
10 HOME : GR
20 FOR S = 1 TO 42: READ P,L
40 POKE 0,P: POKE 1,L: CALL 771: GOSUB 200
50 NEXT S
99 END
100 DATA 192,80,192,80,128,80,128,80,114,80,114,80,128,150,144,80,
    144,80
110 DATA 152,80,152,80,171,80,171,80,192,150,128,80,128,80,144,80,
    144,80
120 DATA 152,80,152,80,171,150,128,80,128,80,144,80,144,80,152,80,
    152,80,171,150
130 DATA 192,80,192,80,128,80,128,80,114,80,114,80,128,150,144,80,
    144,80,152,80,152,80
140 DATA 171,80,171,80,192,150
200 C = C + 1: IF C = 16 THEN C = 1
210 COLOR= C
220 FOR T = 1 TO 100: NEXT T
230 HLIN 30,32 AT 5
240 VLIN 4,6 AT 31
250 HLIN 23,25 AT 32
260 VLIN 31,33 AT 24
270 HLIN 10,12 AT 15
```

```
280 VLIN 14,16 AT 11
290 RETURN
```

Challenge Problems

1. What do lines 200 and 210 do?

2. What do P and L stand for in lines 20 and 40?

3. Why is there an END in the middle of the program?

4. What line would you have to change if you wanted the stars to blink faster or slower?

5. How would you change the program if you wanted the stars to be completely white?

Challenge Programs

1. Take the mystery or adventure game from the previous lesson and add graphics to it. Use both lo-res and hi-res graphics.

Pictures and Shape Tables

This is a difficult subject but once you've created a picture using a shape table you will be richly rewarded with movable pictures. First we will show you how to move and manipulate a shape, then move on to the more difficult subject of how to generate a shape.

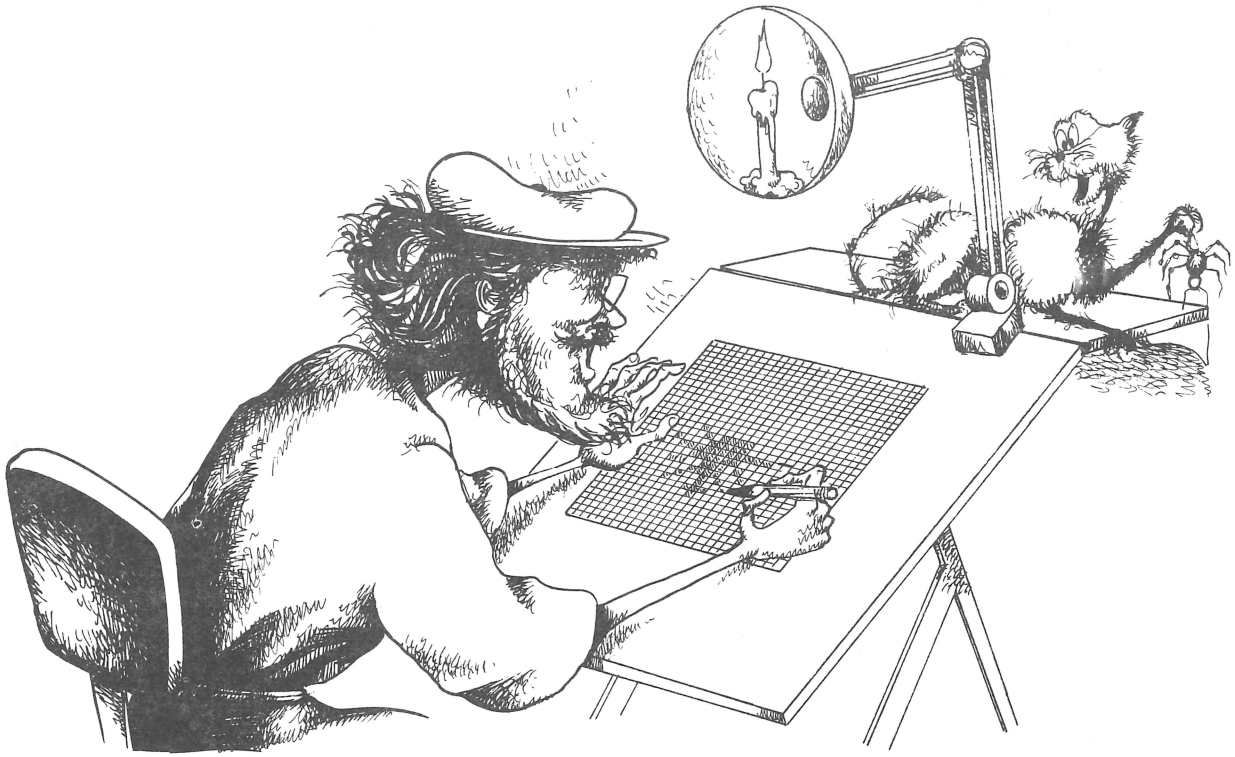
There is a lot to remember. Some of the rules may seem arbitrary, but once you have gained an understanding of the entire concept you will be in a position to better understand the workings of your computer. The following program draws a bug and moves it around the screen.

```
10 REM BUG LOAD
20 FOR A = 900 TO 918
30 READ D: POKE A,D
40 NEXT A
50 DATA 1,0,4,0,40,37,157
60 DATA 54,55,56,6,56,6
70 DATA 56,36,124,42,5,0
100 REM SET UP BUG
110 POKE 232,132:POKE 233,3
120 X = 141:Y = 80
130 R = 0:S = 5:C = 3
140 SCALE= S: ROT= R
150 HGR2 : HCOLOR= C
200 REM COLOR BUG
210 FOR A = 1 TO 3
220 FOR C = 1 TO 7
230 HCOLOR= C
240 DRAW 1 AT X,Y
250 FOR W = 1 TO 200: NEXT W
```

```
260 NEXT C
270 NEXT A
300 REM GROW BUG
310 FOR S = 1 TO 35
320 SCALE= S
330 XDRAW 1 AT X,Y
340 FOR W = 1 TO 50: NEXT W
350 XDRAW 1 AT X,Y
360 NEXT S
370 HGR2 : SCALE= 5
400 REM ROTATE BUG
410 FOR R = 1 TO 64
420 ROT= R
430 XDRAW 1 AT X,Y
440 FOR W = 1 TO 50: NEXT W
450 XDRAW 1 AT X,Y
460 NEXT R
500 REM WALK BUG
510 FOR X = 10 TO 150
520 XDRAW 1 AT X,X
530 FOR W = 1 TO 10: NEXT W
540 XDRAW 1 AT X,X
550 NEXT X
560 DRAW 1 AT X,X
```

Lines 10 through 70 take a series of values from the DATA list and POKE them in memory locations 900 to 918. In the POKE command the A is for address and the D is for data. In the next section we'll explain what the values mean.

Lines 100-150 set up a few housekeeping items so the shape can be drawn. Line 110 sets up the address where the shape can be found. As mentioned, the values used in the address will be explained in the next section. Lines 120 and 130 set up some values to be used later. Notice that they're in the center of the hi-res screen. HGR2 is used for full-screen graphics.



The size of the picture is set by the SCALE= nn command, where nn is a number between 1 and 255. If 5 is chosen, the scale is 5 times the smallest size. The shape can be rotated with the ROT= nn command. Rotation values from 0 to 63 will rotate the figure from zero to 360 degrees in increments of 4 degrees. To take advantage of all 64 angles, set the scale to at least 5. If the scale is set to 1 then a 90-degree rotation will occur at R=16, R=32, R=48 and R=0.

In lines 200-270 all the colors are shifted through three times. Note the use of C in both the FOR/NEXT and HCOLOR commands.

In lines 300-360 the bug is expanded in size. Note the use of R in both the FOR/NEXT and ROT commands. Note also the use of XDRAW. If the color is white, XDRAW will redraw in black. This is equivalent to erasing the picture. Interestingly enough, XDRAW will change violet to

green, orange to blue, green to violet, blue to orange, and black to white! HGR2 is used to reset the screen.

The bug is rotated in lines 400-460, and then moved diagonally in 500-560.

Challenge Problems

1. Change the scale and rotation values to see what happens.

2. Try combining the color and grow sections so the bug goes through all the colors then jumps to the next larger size and goes through all the colors again.

3. Now combine the color, grow, and rotation sections.

Creating Shapes Tables

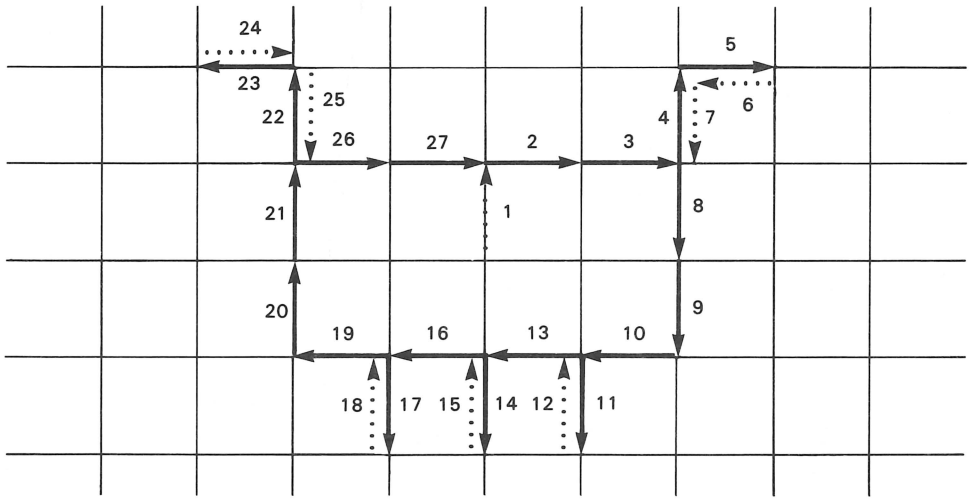
The process of creating shape tables is somewhat involved. Along the way you will probably learn some new math. You will also learn about some secret places inside the Apple computer. If you are willing to explore, the rewards will be great.

You will need several sheets of ¼ inch graph paper as well as lots of scratch paper. Don't touch your computer until everything is planned out on paper.

First, draw a very simple figure like a box or cross using the lines of the graph paper. Now imagine you are in the center of the figure, and you want to walk along all the lines of the figure. You want to start in the center so you can move or rotate the figure naturally. Sometimes you will want to "walk" somewhere without drawing a line or to retrace your steps.

Now draw a solid arrow on the figure for every line you want drawn and a dotted arrow for every line you don't want drawn.

Bug Face



Now number each line in order, from start to finish, solid or dotted. This shape has 27 lines.

The next step is to make a list of all the arrows showing their orientation and type. Use the following table.

Move Table

Arrow	Move Type	Binary Code
	no plot, up	000
	no plot, right	001
	no plot, down	010
	no plot, left	011
	plot up	100
	plot right	101
	plot down	110
	plot left	111

Now refer to the binary code of the Move Table and insert the appropriate code into the binary code column of the List Table below. The first sixteen have already been done. Complete the list and check your answers in the Answer Key at the back of the book.

List Table

Move Number	Arrow	Binary Code	Move Number	Arrow	Binary Code
1		000	14		110
2		101	15		000
3		101	16		111
4		100	17		
5		101	18		
6		011	19		
7		010	20		
8		110	21		
9		110	22		
10		111	23		
11		110	24		
12		000	25		
13		111	26		
			27		

Note: All no-plot move codes start with a zero (0), and all plot moves start with a one (1).

Next the numbers should be arranged into bytes which the computer can read. Each zero (0) or one (1) in the code is called a bit. The computer can process a byte of eight bits at a time. This is the reason many computers are called 8-bit machines.

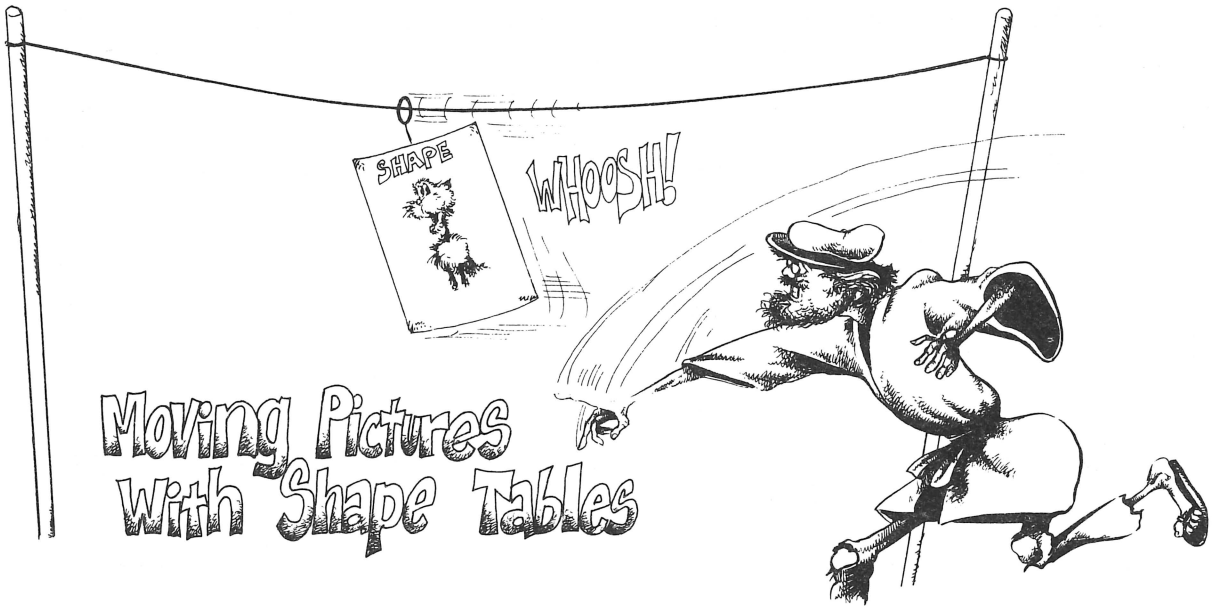
Examine the following Conversion Table carefully.

Conversion Table

Row	3rd Section			2nd Section			1st Section			Decimal
	bit	7	6	5	4	3	2	1	0	
1		0	0	1	0	1	0	0	0	40
2		0	0	1	0	0	1	0	1	37
3		1	0	0	1	1	1	0	1	157
4		0	0	1	1	0	1	1	0	54
5		0	0	1	1	0	1	1	1	55
6		0	0	1	1	1	0	0	0	56
7		0	0	0	0	0	1	1	0	6
8		0	0	1	1	1	0	0	0	56
9										
10										
11										
12										
13										
14										
15										

Note 1: Put only 01, 10, or 11 in the third section, otherwise 00. These values all indicate a move-without plot.

Note 2: If the third and second sections are all zeros put 000 in the 1st section of the following row.



Notice that the third section can hold only a two-bit number, or only binary codes that start with a zero. That means only a 01, 10, or 11 can be placed in the third section. There is one other rule: if zeros are placed in the second and third sections of one byte because you want a no-plot move-up (0 0 0) in the second section and there is a 00 in the third section, the computer will interpret all those zeros as a no-plot no-move. This means that your no-plot move-up instruction must be repeated in the first section of the following byte (with a 0 0 0 entry). Now begin placing the binary codes in the sections beginning with the first section of the top byte, the second section, the third, the first of the next byte, and so forth.

For example, move 1 (000) and move 2 (101) go in byte one, but move 3 (101) won't fit because the lead digit is a one. Therefore, it must be placed in the 1st section of row 2. Move 7 does fit into the third section of row 3 because the lead digit is a zero. But the 12th move (000) can't go into the third section (it isn't a 01, 10, or 11) so it goes into the 1st section of row 7. Note the 15th move. It is a no-plot move-up (000). As the 16th move won't fit into the third section, the third section must be filled with zeros. That makes both the 2nd and 3rd sections zeros so the no-plot move-up must be placed in the 1st section of row 8. To end the table make a row with all zeros. The first eight rows are done for you. See if you can complete the rest of the rows. Check your answers in the Answer Key.

Refer to the following Two's Table to calculate the decimal values needed for the data statements.

Two's Table

7	6	5	4	3	2	1	0
128	64	32	16	8	4	2	1

Now take a look at each row in your conversion table. Match the eight boxes in each row with the numbers in the two's table. Where there is a one, write down the corresponding number from the table above. Look at the following examples:

	7	6	5	4	3	2	1	0	
Row 1	0	0	1	0	1	0	0	0	
	0 +	0 +	32 +	0 +	8 +	0 +	0 +	0	= 40
Row 2	0	0	1	0	0	1	0	1	
	0 +	0 +	32 +	0 +	0 +	4 +	0 +	1	= 37
Row 3	1	0	0	1	1	1	0	1	
	128 +	0 +	0 +	16 +	8 +	4 +	0 +	1	= 157

Note that the decimal numbers above are in the range 1-128. Each of these is a power of 2 (hence, "Two's Table"). If you raise two to the power of the bit position, you can see the decimal value that should be placed there. Remember that in algebra, two to the power of zero is one. Now write the rest of the decimal equivalents necessary to complete the decimal column of the conversion table. Check your answer with the answer key at the back of the book.

Four numbers need to head up the list. The first number tells you how many shapes are in the table. The second is unused and is usually set to zero. The third and fourth tell the number of values that come before the first number in the shape table, in this case 4. Therefore, the lead numbers would be 1, 0, 4, 0, followed by the 15 numbers in the shape table for a total of 19 values. If you had two shapes in the table the fifth and sixth numbers would tell you how many numbers come before the second table. That means 6 places come before the first table so the sequence for two shapes becomes 2, 0, 6, 0, 19, 0, first shape . . . , second shape.

Now we need to tell the computer where to find the beginning of the program. There is a small section of unused memory between 768 and 975. Let's choose 900 for the starting location of the program. We will need two bytes to describe the start number. Refer to the following two-byte table.

Two-byte Two's Table

15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
32768	16384	8192	4096	2048	1024	512	256	128	64	32	16	8	4	2	1
0	0	0	0	0	0	1	1	1	0	1	0	0	1	0	0
0 +	0 +	0 +	0 +	0 +	0 +	512+	256	128+	0+	0+	0+	0+	4+	0+	0+ = 900

7	6	5	4	3	2	1	0
128	64	32	16	8	4	2	1
0	0	0	0	0	0	1	1
0 +	0 +	0 +	0 +	0 +	0 +	2 +	1 = 3

In the computer, 900 takes up two bytes of 8 bits each so we need to break down the number into the left (high) and the right (low) byte. Look at the two-byte 2's table above. Find the largest number that will subtract from 900 and yet leave a positive answer. Subtract that number from 900, and put a "1" under the number. Write the corresponding power of 2's number, 512, under the "1." From the answer to the previous part, find the next largest number, subtract it and put another "1" under it in the table. Write down its corresponding number, 256. If you are not able to subtract a number, put a zero under it. Now divide the two-byte table into the left half and the right half. Add up the values for the right half (128 + 4 = 132). This number becomes the low order byte. Next, imagine that the right half is placed over the left half so the 0 falls over the 8, the 1 over the 9, etc. Now write down the new corresponding powers of 2's numbers and add them up (2 + 1 = 3). This is the high order byte.

The low order byte must be poked into memory location 232 and the high into 233. In our example, we would use:

```
POKE 232,132: POKE 233,3
```

As an alternative to using the FOR/NEXT and READ/DATA commands the shape table can be saved in a binary file once the above program is run. Use the following, where B indicates that the file is to be saved as a binary file, A is the decimal address in memory where the file will start when loaded from the disk, and L is the length of the file.

```
BSAVE BUG,A900,L19
```

Once the binary file has been saved, the FOR/NEXT and READ/DATA statements can be eliminated from a program and the file can be called up by entering:

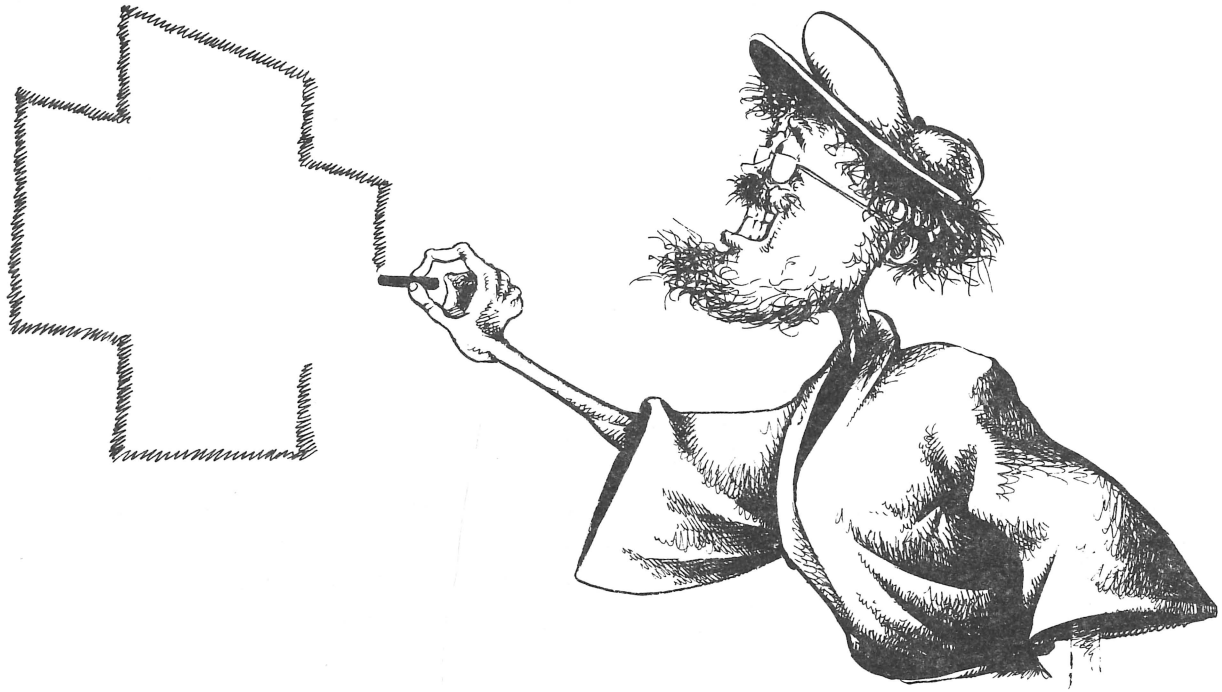
```
PRINT CHR$(4);"BLOAD BUG"
```

Follow that by POKEs to set the start address so the graphics commands can find the table, set the scale, and then draw shape 1 with the command:

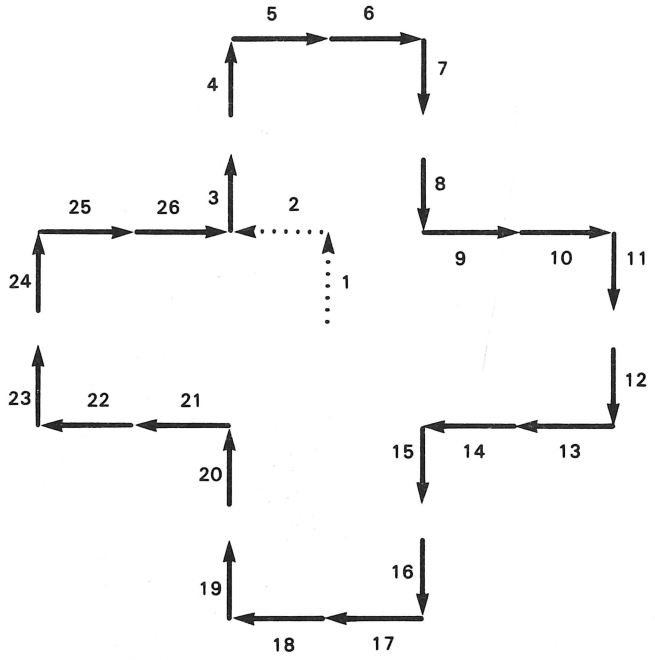
```
DRAW 1 AT X,Y
```

Drawing a Cross

Now see if you can draw a cross using shape tables.



The cross could be numbered as follows.

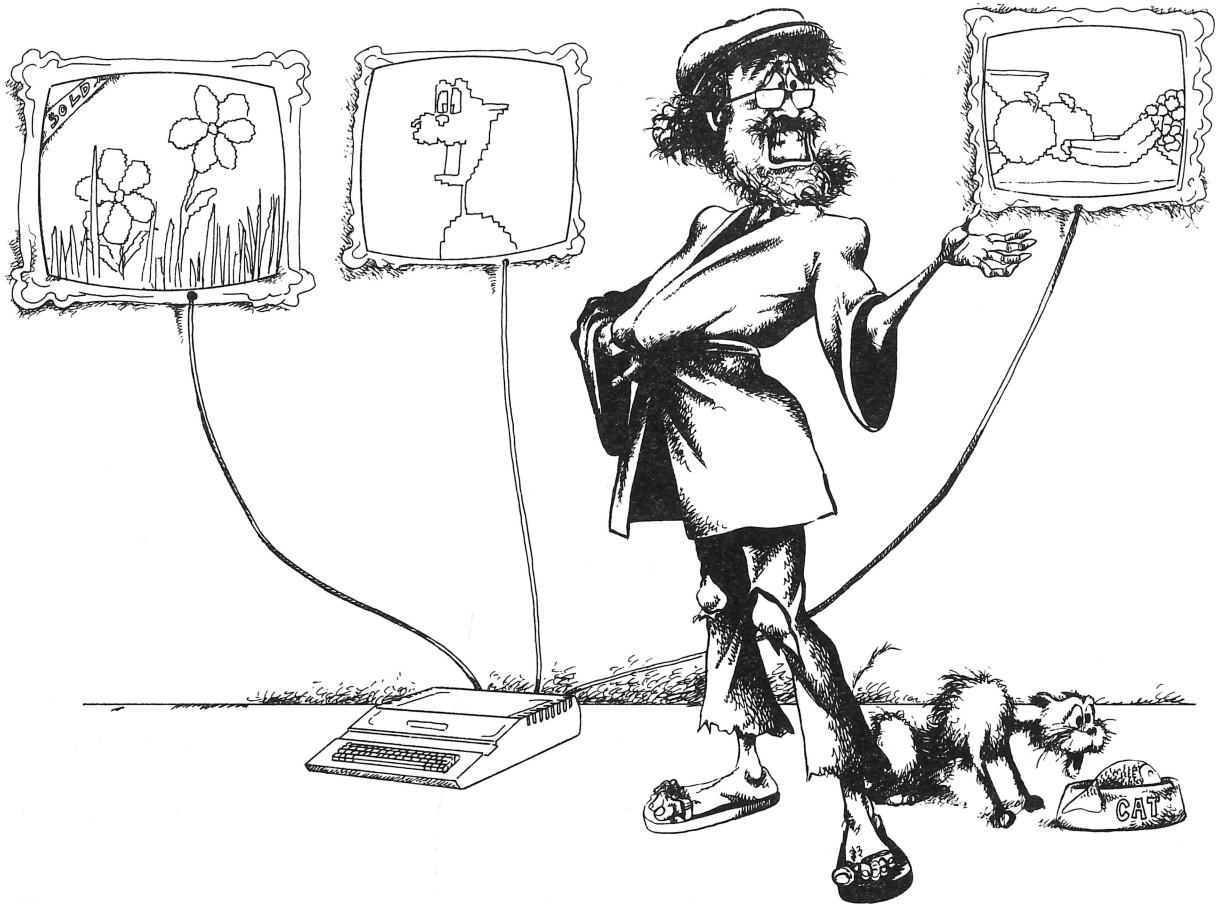


Once assembled, the shape table should yield 15 numbers. Together with the four lead numbers there should be 19 numbers in all, provided you followed the path diagrammed above. Use the numbers in a FOR/NEXT loop and READ/DATA program as was done in the previous section, "Pictures and Shape Tables." You might also want to try moving the cross around the screen.

Once the program is run the values are in memory. They can be saved into a binary file with the following command:

```
BSAVE CROSS,A900,L19
```

See the answer key for a way to solve the problem.


SOLD
CAT

Answer Key for Problems and Programs

There is no one correct answer to most of the challenge programs. There are many answers which will work and therefore are equally "correct." The purpose of this answer key is to show you one way to solve the question.

Programming Tips and Techniques

Problems, page 17

1. Here is how you calculate the number:

   14/2 = 7 and 20 – 7 = 13 So your command is HTAB 13

2. 10 HOME
   20 PRINT "WELCOME TO <YOUR NAME>'S DISK"
   30 VTAB 7 : PRINT "TYPE IN 'CATALOG' TO SEE THE PROGRAMS"

Low Resolution Graphics

Programs, page 23

1. 10 GR
   20 COLOR= 9
   30 PLOT 39,39

   For line 30, any two numbers that are both greater than 20 and less than 39 will do.

2. 10 GR
   20 COLOR= 13
   30 PLOT 20,20

   Again, for line 30 any pair of numbers are acceptable as long as they are within the screen dimensions (0-39,0-39).

GOTO Loops — IF/THEN

Problems, page 28

1. 10 GR : COLOR= 6
   20 B = 19
   30 C = 15
   40 PLOT B,C
   50 B = B + 1 : C = C + 1
   60 IF B = 39 THEN END
   70 GOTO 40

   Any two numbers where B > C will work for lines 20 and 30.

2. Change line 10 to read:

   10 GR : COLOR= X (Where X is any number between 1 and 15 other than 6).

3. Change line 50 to read:

   50 B = B + 2 : C = C + 2

   Note: If you get an ?ILLEGAL QUANTITY ERROR it simply means you went off the screen. In line 60 just change B = 39 to B > 39.

4. Change line 60 to read:

```
60 IF B > 20 THEN END
```

5. It made a "beep" sound.

VLIN and HLIN

Programs, page 30

1.
```
10 GR
20 COLOR= 5
30 VLIN 18,22 AT 19
40 HLIN 17,21 AT 20
```

2.
```
10 GR
20 COLOR= 5
30 VLIN 18,22 AT 19
40 COLOR= 10
50 HLIN 17,21 AT 20
```

3.
```
10 GR : COLOR= 6
20 REM EYES
30 HLIN 12,14 AT 9
40 HLIN 12,14 AT 11
50 PLOT 12,10
60 PLOT 14,10
70 HLIN 19,21 AT 9
80 HLIN 19,21 AT 11
```

```
90 PLOT 19,10
100 PLOT 21,10
110 REM RIGHT EYEBROW
120 PLOT 12,5 : PLOT 13,6 : PLOT 14,7
130 REM LEFT EYEBROW
140 PLOT 19,7 : PLOT 20,6 : PLOT 21,5
150 REM MOUTH
160 HLIN 13,20 AT 18
170 PLOT 12,17 : PLOT 11,16 : PLOT 21,17 : PLOT 22,16
```

4. Change line 10 to read:

   10 GR : COLOR= 7 (You may also use 2 or 14.)

   Add line 115 as follows:

   115 COLOR= 8

   Add line 155 as follows:

   155 COLOR=1

Area of a Rectangle in Graphics

Problems, page 32

1. The program prints the message after THEN and goes back to line 20.

2. Because both the length and width of the graphics page is 40, but the columns and rows are numbered 0-39. If you tried to PLOT 40,40 you would get an ?ILLEGAL QUANTITY ERROR.

Background Colors

Problems, page 34

1. REM is the abbreviation for REMark.

2. When the computer sees the REM it ignores everything that follows on that line and goes on to the next line.

Programs, page 34

1. Change line 20 to read:

   20 GR : COLOR= 6 (You may also use 2, 7, or 14)

2. Add a line at 45 to read:

   45 FOR T = 1 TO 500 : NEXT T

3. Change line 40 to read:

   40 VLIN 0,39 AT RW

4. Change line 30 and 40 to read:

   30 FOR RW = 0 TO 19
   40 HLIN 0,39 AT RW (changed back from VLIN)

Then add the following lines:

   55 COLOR=12 (You may also use 4)
   60 FOR RW = 20 TO 39

```
70 HLIN 0, 39 AT RW
80 NEXT RW
```

5. Add the following lines for a cross:

```
90 COLOR= 0
100 VLIN 14,19 AT 19
110 HLIN 17,21 AT 16
```

Or use the following for a rectangle:

```
90 COLOR= 0
100 HLIN 19,21 AT 17
110 HLIN 19,21 AT 18
120 HLIN 19,21 AT 19
```

6. Change the following lines to read:

```
20 GR
30 FOR RW = 0 TO 39
34 C = C + 1
36 COLOR= C
40 HLIN 0,39 AT RW
45 IF C=15 THEN C=1
50 NEXT RW
```

An alternate method would be:

```
20 GR
30 FOR RW=0 TO 39
35 COLOR= RW
40 HLIN 0,39 AT RW
50 NEXT RW
```

For variety try:

```
20 GR
30 FOR RW = 0 TO 39
35 FOR C = 1 TO 15
40 HLIN 0,39 AT RW
42 COLOR= C
43 FOR T=1 TO 200:NEXT T
45 NEXT C
46 NEXT RW
```

And try:

```
20 GR
30 FOR C=1 TO 15
35 COLOR= C
40 FOR RW=0 TO 39
42 HLIN 0,39 AT RW
45 NEXT RW
50 NEXT C
```

Moving Graphics

Problems, page 37

1. Set the background color.

2. So you can print the word HI below the graphics screen in the text window (bottom 4 lines).

3. Draws the tongue in the same color as the background color, effectively "erasing" the tongue.

4. It is a counter to have the tongue stick out 50 times, then stop.

Programs, page 37-38

1. Change lines 10, 100, and 310 to the following:

```
10 HOME : GR : COLOR= 14
100 COLOR= 1
310 COLOR= 13
```

2. Change lines 320 and 340 to the following:

(faster)
```
320 FOR W = 1 TO 50 : NEXT W
340 FOR W = 1 TO 50 : NEXT W
```
(any number less than 100 is acceptable)

(slower)
```
320 FOR W = 1 TO 500 : NEXT W
340 FOR W = 1 TO 500 : NEXT W
```
(any number more than 100 is acceptable)

3. Change line 350 to the following:

```
350 C = C + 1 : IF C < 25 THEN GOTO 310
```

4. You should have changed the "HI" in line 200 to something else.

5. Here's a possible solution for the box:

```
10 HOME : GR : COLOR= 8
20 HLIN 20,30 AT 20
30 VLIN 20,30 AT 20
40 HLIN 20,30 AT 30
50 VLIN 20,30 AT 30
55 REM WORM
60 COLOR= 12
70 HLIN 27,29 AT 12
80 PLOT 28,13
```

```
90 HLIN 27,29 AT 14
100 HLIN 26,28 AT 15
110 HLIN 26,28 AT 16
120 HLIN 25,27 AT 17
130 HLIN 25,27 AT 18
140 HLIN 24,26 AT 19
150 REM EYES
160 COLOR= C
170 PLOT 27,13: PLOT 29,13
180 FOR T = 1 TO 500: NEXT T
190 C = C + 1
200 IF C / 2 = INT (C / 2) THEN COLOR= 0: GOTO 70
210 GOTO 60
```

Note: In line 200 the computer checks to see if C is even. If it is, the color is set to black and the worm is redrawn, effectively erasing it.

There are other ways of achieving the illusion of motion. You'll notice that a part of the worm doesn't get "erased." This is done deliberately to demonstrate other effects that can be used in animation.

This program is in a continuous loop. You may use a FOR/NEXT loop or an IF/THEN structure to limit the number of times the worm "pops."

High Resolution Graphics

Problems, page 40

1. TEXT

2. High resolution GRaphics

3. High resolution **COLOR**

4. No, these commands are only available in lo-res graphics.

Programs, page 41

1. 10 HGR : HCOLOR=3
   20 HPLOT 20,17 TO 100,17 TO 100,70 TO 20,70 TO 20,17

2. 10 HGR : HCOLOR= 5
   20 HPLOT 130,0 TO 240,120 TO 20,120 TO 130,0
   30 HPLOT 20,30 TO 240,30 TO 130,160 TO 20,30

3. The following program is one of many possible solutions:

```
10 HGR : HCOLOR= 3
20 FOR X = 0 TO 279 STEP 20
30 FOR Y = 0 TO 159 STEP 20
40 HPLOT X,0 TO 0,Y
50 NEXT : NEXT
60 FOR X = 279 TO 0 STEP -20
70 FOR Y = 159 TO 0 STEP -20
80 HPLOT X,159 TO 279,Y
90 NEXT : NEXT
```

More Hi-Res Problems, page 42

1. Answers will vary. The following program will draw a circle which you can use for the sun in your picture.

```
10 HGR : HCOLOR= 3
20 PX = 140:PY = 80:R = 50
```

```
30 HPLOT PX + R,PY
40 FOR D = 0 TO 6.3 STEP 0.1
50 X = PX + R * COS (D)
60 Y = PY + R * SIN (D)
70 HPLOT TO X,Y
80 NEXT D
```

Lines 50 and 60 above contain the two equations that actually draw the circle. In line 20, PX and PY determine where the center should be, and R is the radius. Line 30 HPLOTs the starting point, and line 70 HPLOTs to the next point on the circle as calculated in lines 50 and 60. The FOR/NEXT loop goes around the circle in half-degree increments.

To draw a sun in the upper right-hand corner of the screen change line 20 to the following:

```
20 PX = 250:PY = 30:R = 25
```

2. Answers will vary.

Three Dimensional Graphics

Problems, page 50

1. Add line 200 to draw a seagull:

```
200 HPLOT 10,10 TO 12,12 TO 14,10
```

2. Answers will vary.

3. Answers will vary.

A-Mazing Graphics

Problems, page 53

1. Answers will vary.

2. Answers will vary.

3. Answers will vary.

Hi-Res Background Colors

Problems, page 56

1. Two

2. Eight, but two are black and two are white, giving a total of six colors from which to choose.

3. 140,80

4. 140,96

Programs, page 56

1. Change line 20 to the following:

```
20 HCOLOR= 2
```

2. Change lines 10 and 20 to the following:

```
10 HGR2
20 HCOLOR= 2
```

3. Color loops:

```
10 HGR
20 FOR C = 0 TO 7
30 HCOLOR= C
40 HPLOT 80,80
50 CALL -3082
60 FOR T = 1 TO 300: NEXT T
70 NEXT C
```

4. Box on blue background:

```
10 HGR : HCOLOR= 6
20 HPLOT 80,80
30 CALL -3082
40 HPLOT 100,80 TO 180,80 TO 180,100 TO 100,150 TO 100,80
```

Hi-Res Action

Problems, page 58

1. Line 70 is a counter. It counts the number of times the worm jumps out of the box (once each time the loop is executed).

2. It sets the screen color to black to redraw (or erase) the worm.

3. Because it is a continuation of line 110. The plot goes from the last HPLOT at 150,35.

Programs, page 58

1. Change line 140 to a number less than 300 to make it faster, or greater than 300 to make it slower.

2. Change line 80 to the following:

```
80 IF C = 20 THEN TEXT : HOME : END
```

3. Make the following changes to the program:

   - Delete line 30 and move line 70, the counter, to line 30.
   - Move line 80 to line 31.
   - Add the following lines to the program:

```
32 FOR CL = 0 TO 7
35 HCOLOR= CL
45 NEXT CL
```

   - Delete lines 70 and 80.

4. Answers will vary.

Alternating Between Graphics Pages One and Two

Problems, page 60

1. Twenty

2. Lines 70, 80, 90, and 100.

More Problems, page 62

1. RETURN

2. Four text lines or 32 high resolution lines

3. None

4. They switch the screen from graphics page one to graphics page two, and vice versa.

5. They "erase" or redraw the subroutine design in the background color.

Programs, page 62

1. Change line 1330 to the following:

```
1330 HPLOT 200,110 TO 250,110 TO 250,120 TO 200,120 TO 200,110
```

2. Change line 10 and add line 20 as follows:

```
10 HGR
20 VTAB 23: PRINT "I'M HUNGRY, LET'S EAT."
```

Note: The bottom of graphics page two will display "garbage."

3. Answers will vary. In this program the subroutine in lines 1110-1130 draw the stationary parts of the picture: the nose, eyes, head, and ear. While drawing on graphics page one, line 60 with GOSUB 1210 "undraws" the open mouth, and line 70 with GOSUB 1310 draws the closed mouth. Lines 150 and 170 REVERSE the procedure while drawing on graphics page two. The program redraws the lines each time it executes the loop. This was done deliberately so that the graphics could be made to move to more than just two positions.

Drawing in Hi-Res with Paddles

Problems, page 64

1. POKE -16302,0 sets the entire screen for hi-res graphics, leaving no area at the bottom for text.

2. Change line 5 and add line 115 to the program:

```
5 HOME : PRINT "WHENEVER YOU WANT TO CLEAR THE SCREEN AFTER
  DRAWING, PRESS ANY KEY ON THE KEYBOARD. IF YOU WISH TO STOP
  THE PROGRAM TYPE 'S.' "
115 IF Z = 211 THEN TEXT : HOME : END
```

3. Change lines 60, 70 and 80 to the following:

```
60 X = PDL (0) * 100/255
70 Y = PDL (1) * 100/255
80 IF X > 0 AND X < 100 AND Y > 0 AND Y < 100 THEN GOTO 100
```

Using Joystick/Paddles to Fire Missiles

Problems, page 66

1. It tells the computer to count backwards by ones from 159 to 1.

2. Line 45 plots it in color 3, and line 60 replots it in black, effectively "erasing" it.

3. Change it to 39/255.

Programs, page 66

1. Change line 35 to the following:

```
35 FOR I = 1 TO 159
```

2. Change the following lines:

```
10 HOME : GR
20 X = PDL (0) * 39/255
35 FOR I = 40 TO 1 STEP -1
40 COLOR= 15
45 PLOT X,I
60 COLOR= 0 : PLOT X,I
```

Shooting Random Targets

Problems, page 68

1. Checks to see if the paddle button has been pressed.

2. Redraws the missile and the target in black (the background color), effectively "erasing" them.

3. Tells the computer to beep the speaker.

Programs, page 68

1. Change line 40 to the following:

```
40 Y = INT ( RND (1) * 40)
```

2. To use only the middle 20 columns of the screen you might try:

```
30 N = 10 + INT ( RND (1) * 20)
```

Sounds Effects for Games

Programs, page 75

1. Once you have found the P and L values you want, delete lines 20, 25 and 45, and put the two values in line 30 in place of P and L.

2. Delete line 40 and change line 35 to the following:

```
35 CALL 848
```

Sound Subroutines

Problems, page 78

1. Three times

2. RETURN

3. Three

Programs, page 79

1. Answers will vary. Use the format found in lines 10-40. Vary the INVERSE, FLASH, bells, and right and wrong answer responses.

2. To shorten the programming one might try the following in place of lines 10-40:

```
10 INPUT "WHAT DOES CRT STAND FOR?";A$
20 IF A$ = "CATHODE RAY TUBE" THEN FLASH : PRINT "CORRECT": NORMAL :
    GOSUB 1000 : GOTO 50
30 PRINT "WRONG": GOSUB 400
```

3. You could put a graphics subroutine at 2000 and insert a GOSUB 2000 after the GOSUB 1000. Just before the RETURN in the graphics subroutine be sure to issue a TEXT command.

Introduction Apple Music

Problems, page 84

1. 50 FOR S = 1 TO 12

2. Tells the computer to activate the sound-making program.

Programs, page 84

1. Answers will vary.

2. Answers will vary.

Note: Be sure there are an even number of values in the DATA statements. Don't use more than six or eight pairs of values in each DATA statement or they become difficult to correct.

Space Music

Problems, page 86

1. Prepares the computer speaker to play music.

2. Has the computer read and play the 19 notes in the tune.

3. The pitch of the first note, an A sharp.

4. The length of the first note, an eighth note.

5. RESTORE causes the computer to start reading DATA statements from the top again, starting with the first DATA statement in the program. That is why line 250 says GOTO 20 instead of GOTO 100. If we started at line 100 the program would read in the DATA on line 40 before it got to the DATA at line 140.

6. So the tune will be played only twice, then END.

Programs, page 86

1. Answers will vary.

Combining Graphics and Sound

Problems, page 88

1. Roll through the sixteen colors in lo-res graphics.

2. Pitch and Length of the notes in the melody.

3. To keep the program from executing the subroutine one more time.

4. Line 220

5. Delete line 200 and have line 210 read:

```
210 COLOR= 15
```

Programs, page 89

1. Answers will vary.

Pictures and Shape Tables

Problems, page 94

1. In line 130 try values of 1 to 255 for S. If you use 0 for S you will get the maximum size for the shape. Try any value from 0 to 64. Values beyond 64 just duplicate the same angles, but cannot exceed 255 or go below 0.

2. Combine sections 200-399 as follows:

```
200 REM COMBINE GROW & COLOR
210 FOR S = 1 TO 35
220 SCALE= S
230 FOR C = 1 TO 7
240 HCOLOR= C
250 XDRAW 1 AT X,Y
260 FOR W = 1 TO 50:NEXT W
```

```
270 XDRAW 1 AT X,Y
280 NEXT C
290 NEXT S
```

3. Combine sections 200-499 as follows:

```
200 REM COMBINE 3
210 FOR S = 5 TO 35
220 SCALE= S
230 FOR C = 1 TO 7
240 HCOLOR= C
250 FOR R = 0 TO 64 STEP 4
260 ROT= R
270 XDRAW 1 AT X,Y
280 XDRAW 1 AT X,Y
290 NEXT R
300 NEXT C
310 NEXT S
```

Creating Shape Tables

Page 97, 98

List Table

Move Number	Arrow	Binary Code	Move Number	Arrow	Binary Code
1		000	14		110
2		101	15		000
3		101	16		111
4		100	17		110
5		101	18		000
6		011	19		111
7		010	20		100
8		110	21		100
9		110	22		100
10		111	23		111
11		110	24		001
12		000	25		010
13		111	26		101
			27		101

Conversion Table

Row	3rd Section		2nd Section			1st Section			Decimal
	bit	7	6	5	4	3	2	1	0	
1		0	0	1	0	1	0	0	0	40
2		0	0	1	0	0	1	0	1	37
3		1	0	0	1	1	1	0	1	157
4		0	0	1	1	0	1	1	0	54
5		0	0	1	1	0	1	1	1	55
6		0	0	1	1	1	0	0	0	56
7		0	0	0	0	0	1	1	0	6
8		0	0	1	1	1	0	0	0	56
9		0	0	0	0	0	1	1	0	6
10		0	0	1	1	1	0	0	0	56
11		0	0	1	0	0	1	0	0	36
12		0	1	1	1	1	1	0	0	124
13		0	0	1	0	1	0	1	0	42
14		0	0	0	0	0	1	0	1	5
15		0	0	0	0	0	0	0	0	0

Drawing a Cross, page 103

```
1 REM CROSS DRAW
10 FOR A = 900 TO 918
20 READ D: POKE A,D
30 NEXT A
40 DATA 1,0,4,0,3,32,44
50 DATA 53,46,53,62,55,62
60 DATA 39,60,39,44,5,0
100 HGR2 : HCOLOR= 3
110 POKE 232,132: POKE 233,3
120 SCALE= 5: ROT= 0
130 X = 140:Y = 80
140 DRAW 1 AT X,Y
```

Once the above program is run, type the following at the prompt (be sure to clear the screen with a TEXT command first):

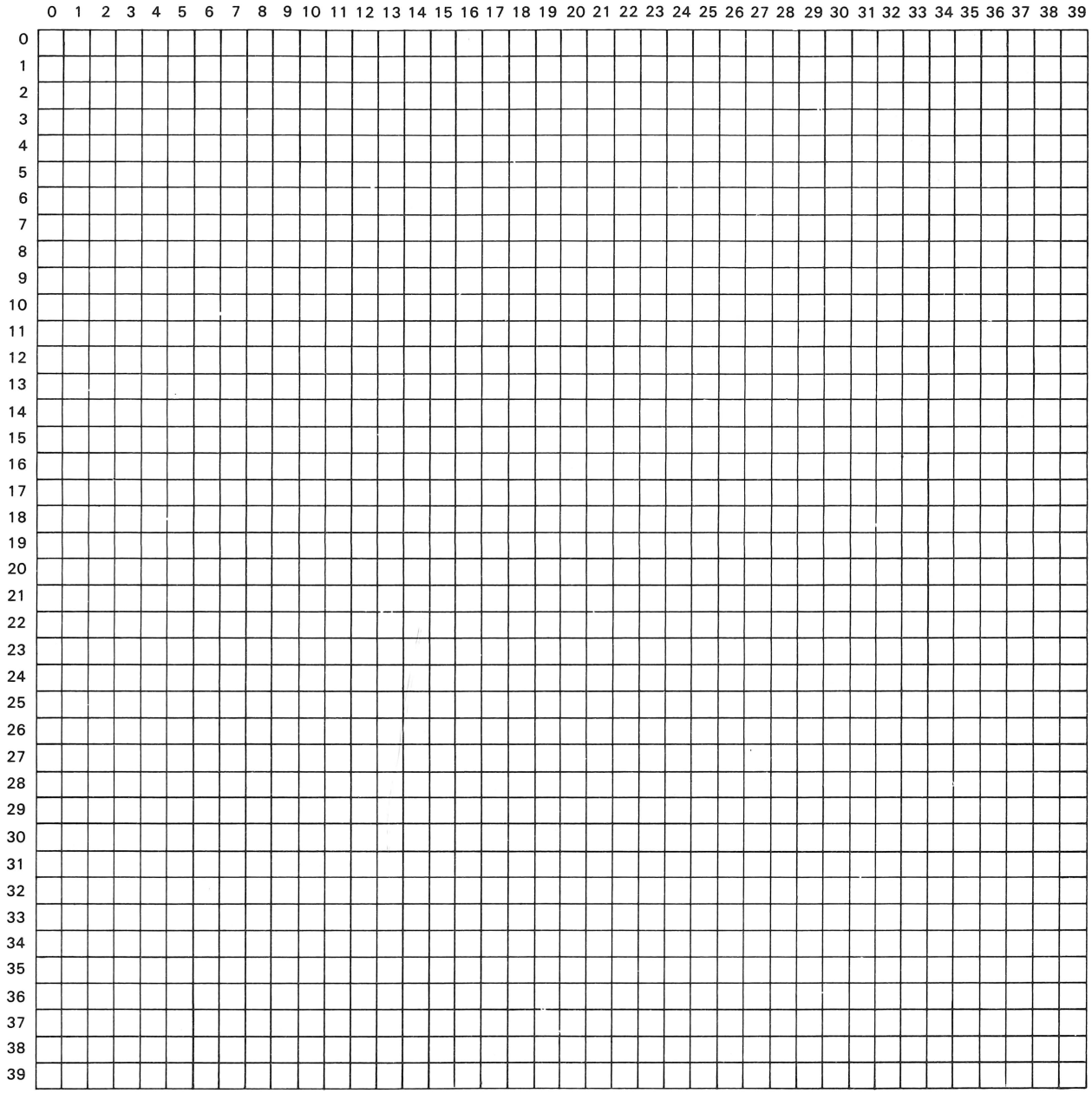
```
BSAVE CROSS,A900,L19
```

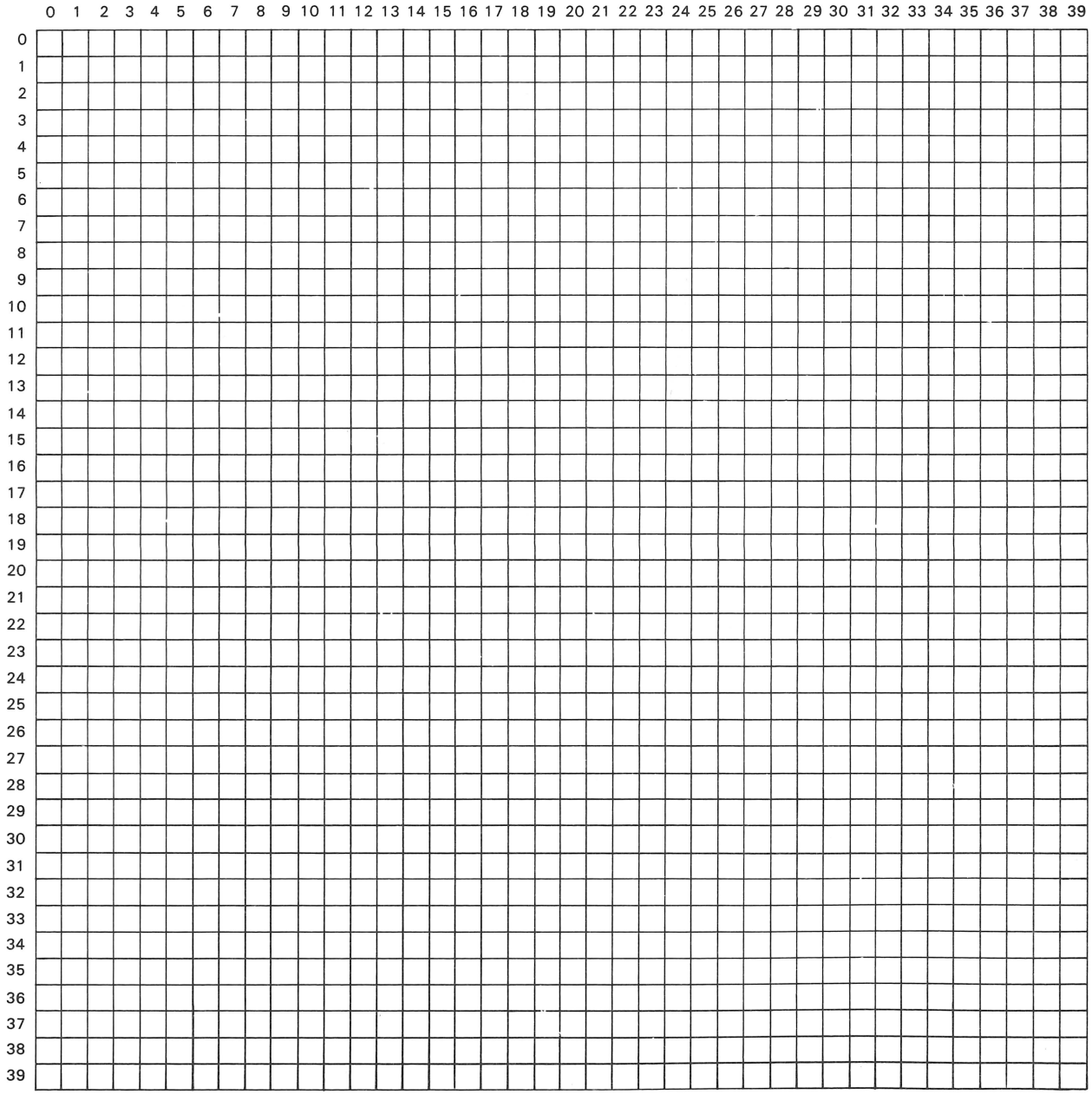
Here is a way to do CROSS MOVES. First be sure CROSS is saved as a binary file by CATALOGing and checking to see if there is a B in front of your CROSS file. Using D$ = CHR$(4) is a way to load the binary file CROSS into memory so it can be used in a program without DATA statements.

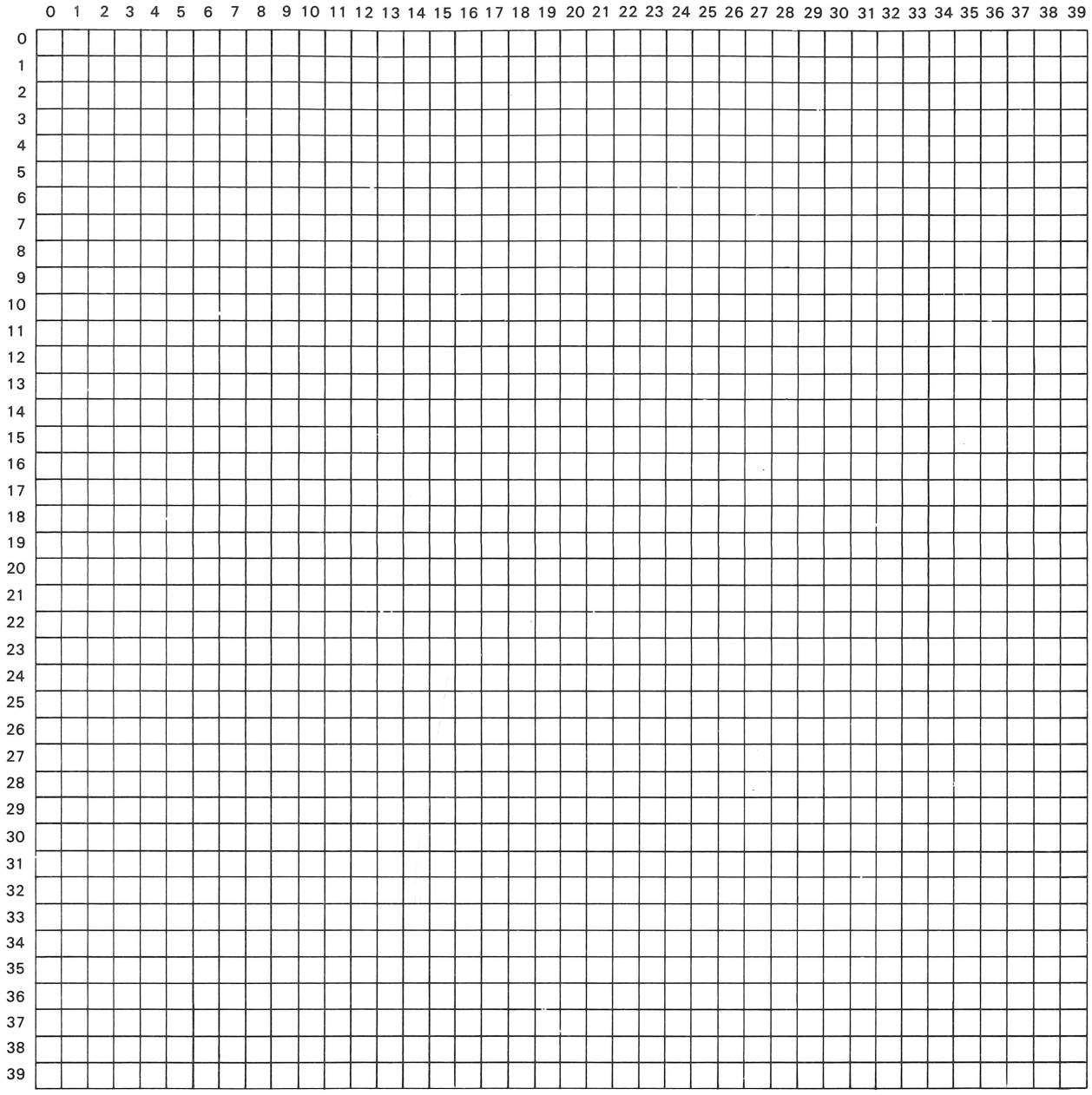
```
10 D$ = CHR$ (4)
20 PRINT D$;"BLOAD CROSS"
100 HGR2 : HCOLOR= 3
110 POKE 232,132: POKE 233,3
120 X = 140:Y = 80
200 FOR S = 1 TO 25
210 SCALE= S
220 XDRAW 1 AT X,Y
230 FOR W = 1 TO 500: NEXT W
```

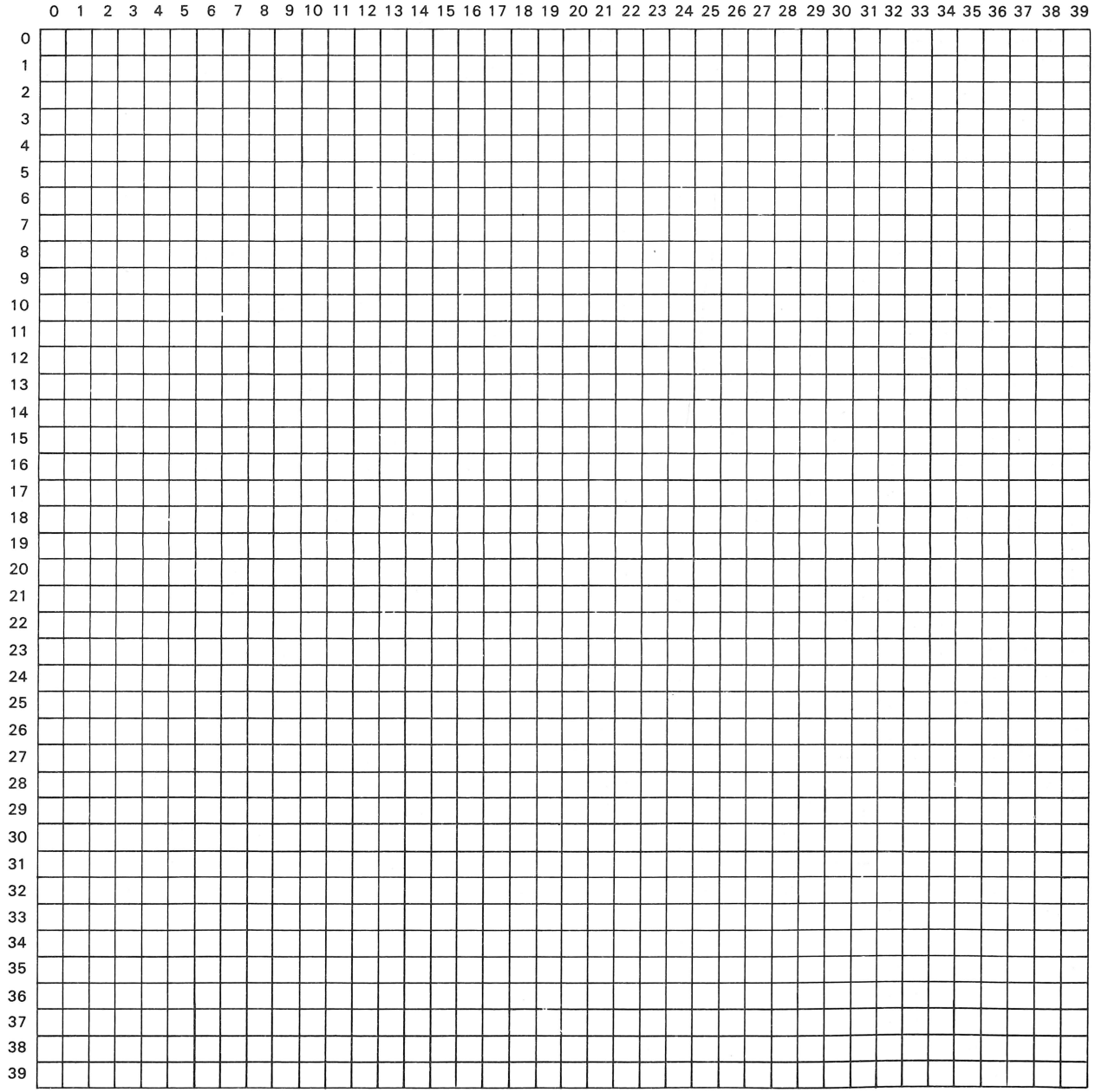
```
240 XDRAW 1 AT X,Y
250 NEXT S
300 SCALE = 10
310 FOR R = 0 TO 63
320 ROT= R
330 XDRAW 1 AT X,Y
340 FOR W = 1 TO 50: NEXT W
350 NEXT R
```

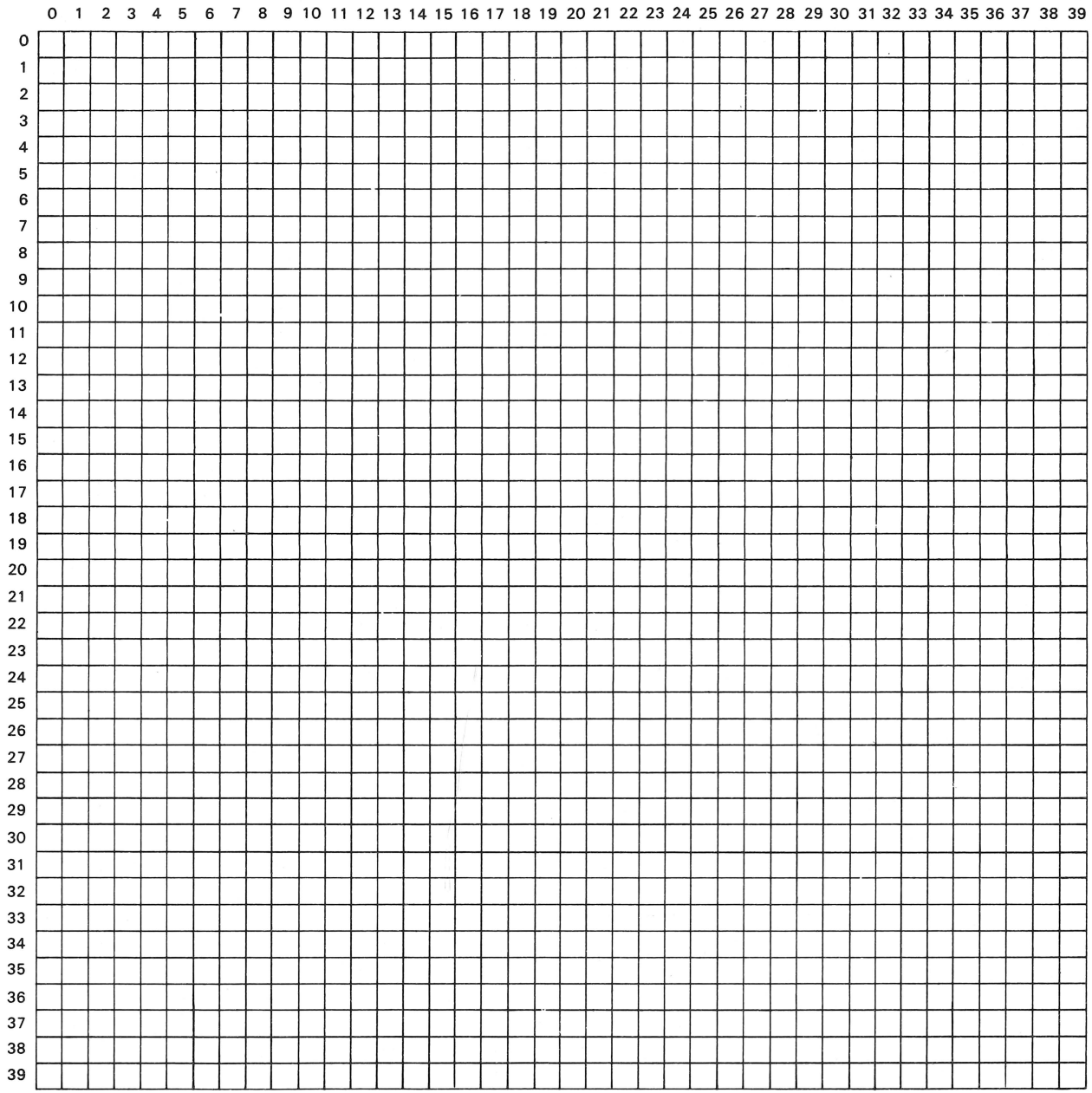
Make some copies of this graph paper to plot designs.

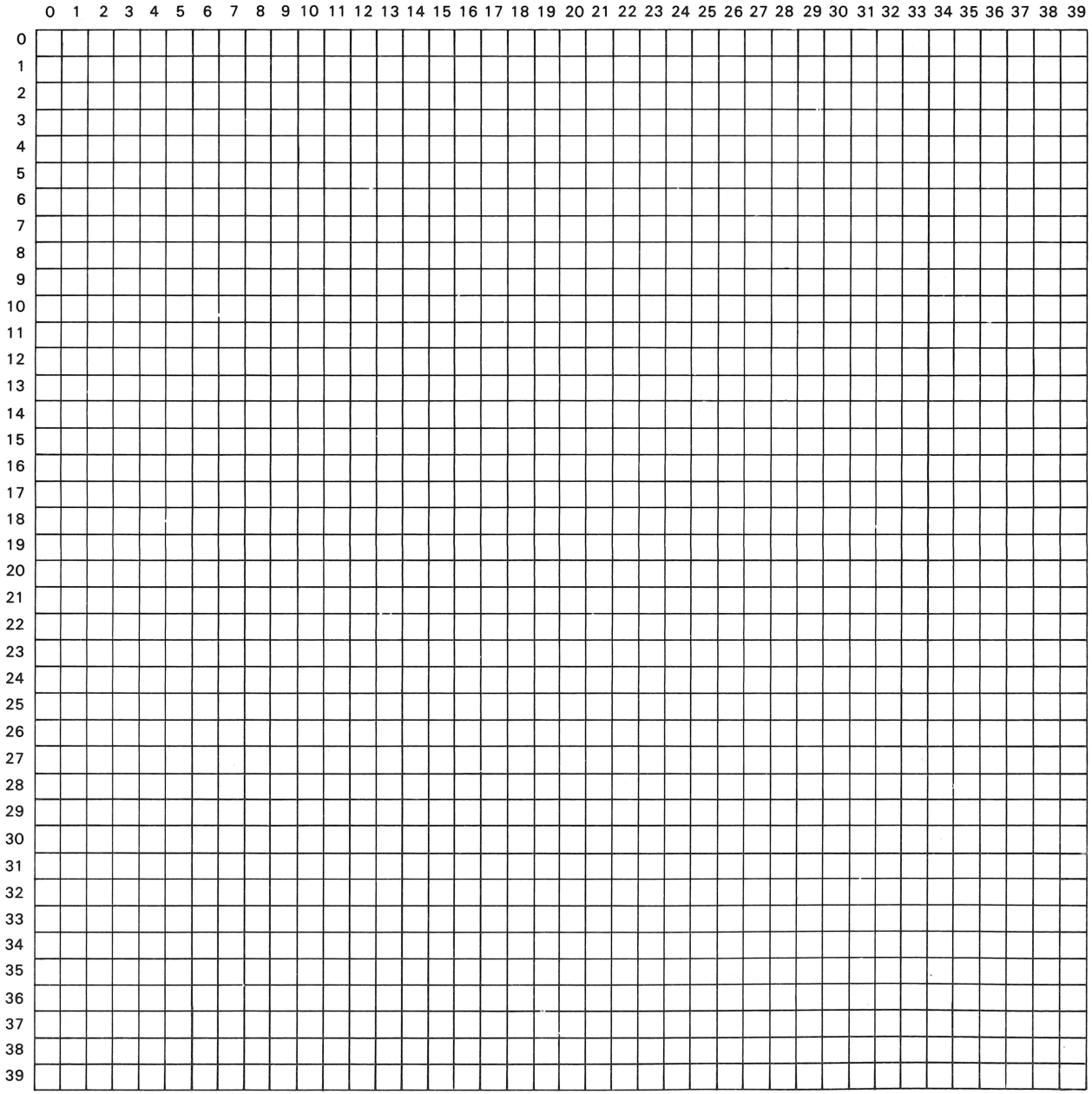


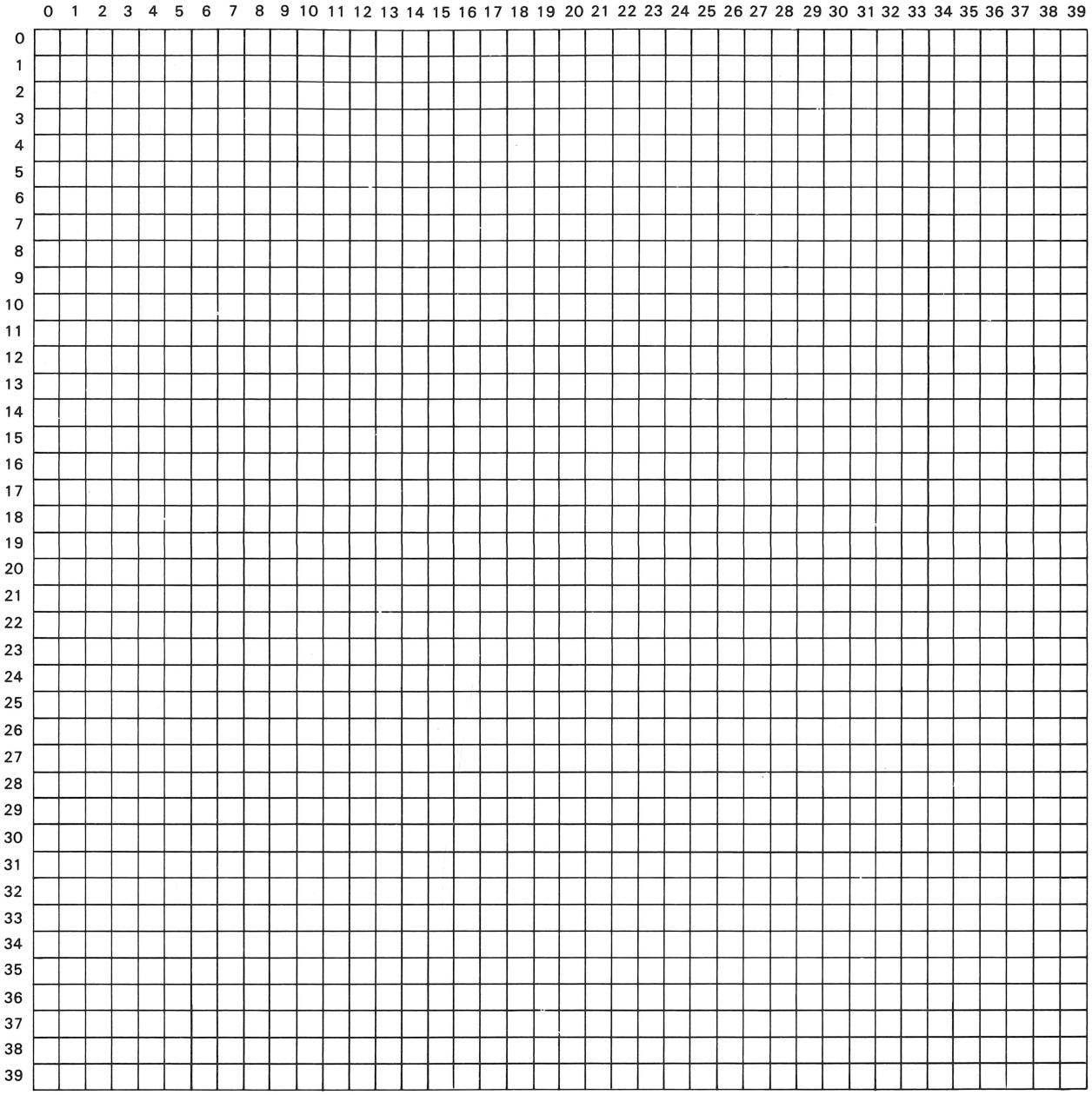

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

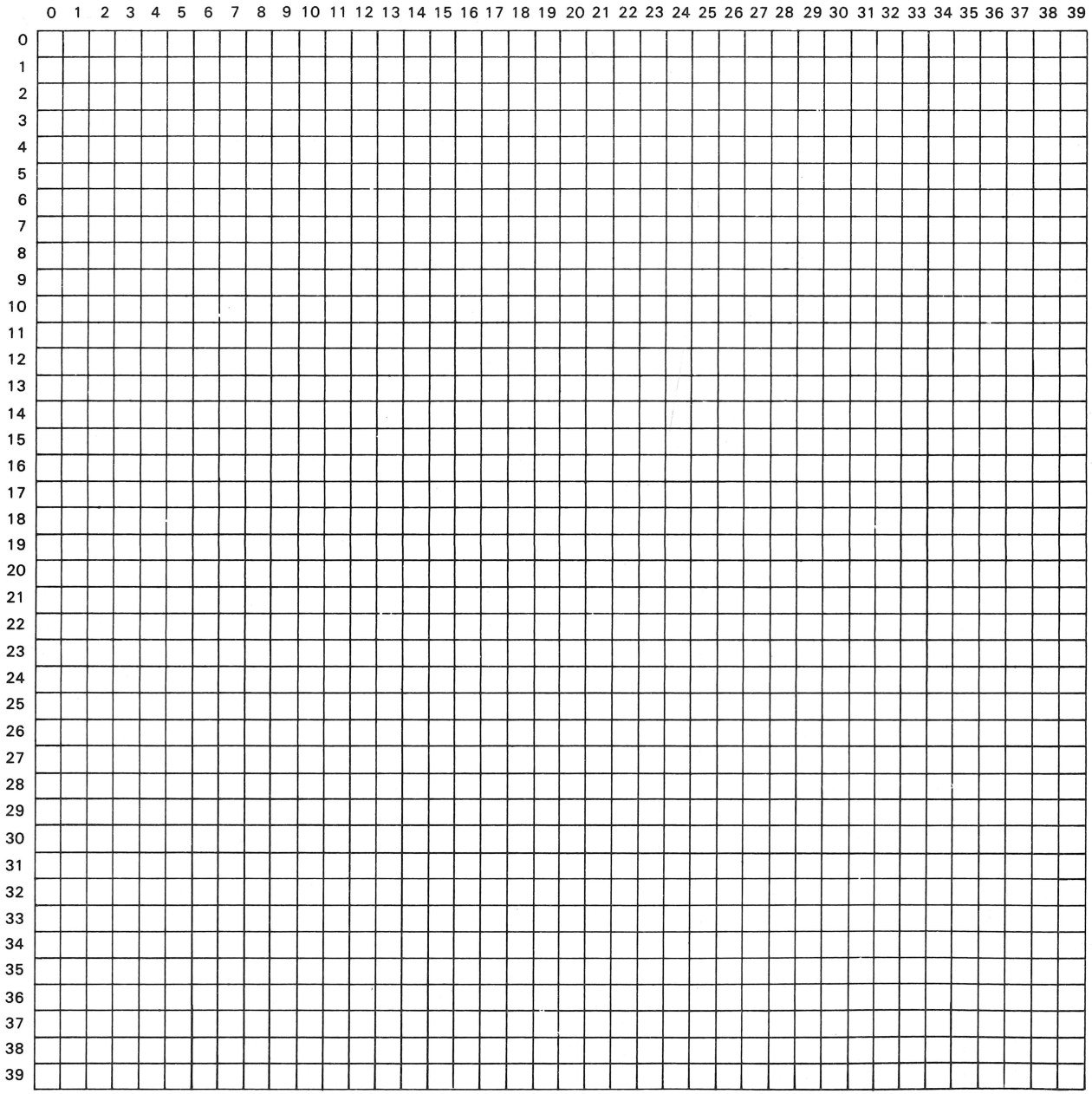

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

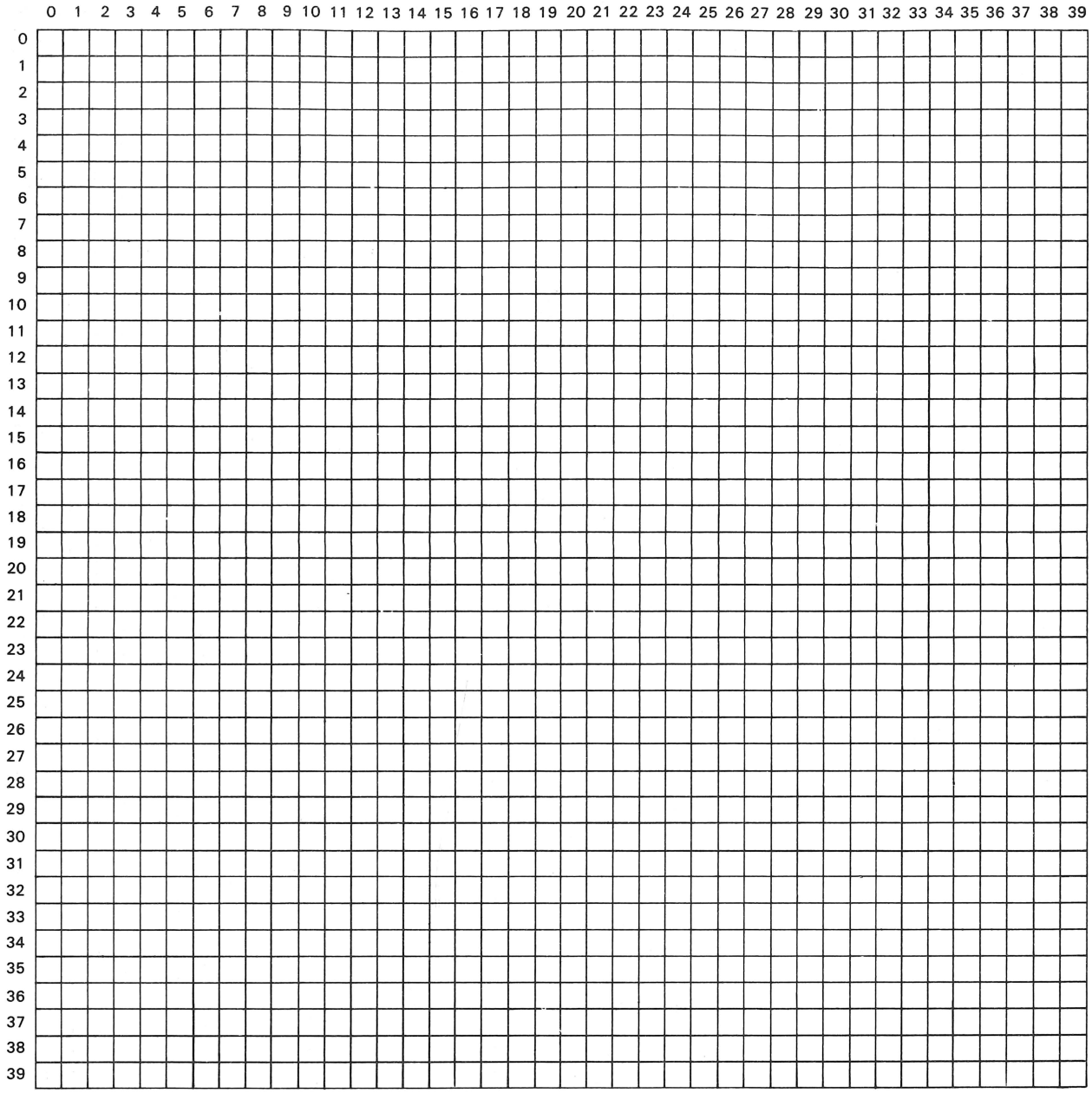

0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39


0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 37 38 39

NOT FOR RESALE

SOUND AND GRAPHICS

Written especially for children in grades two through six, but suitable for ANY beginner, SOUND AND GRAPHICS teaches BASIC graphics techniques on the Apple II + and //e and Franklin computers.

This marvelous book concentrates on sound and graphics, which provides immediate satisfaction and keeps a child's interest. The simple BASIC programs included are arranged step-by-step, with problems and challenges so children can be sure they understand a concept before moving on.

SOUND AND GRAPHICS includes chapters on programming tips and techniques, low and high resolution graphics, drawing with paddles, three dimensional graphics, firing missiles, sound subroutines, Apple music, shape tables and how to combine graphics with sound. A complete Answer Key to the questions and challenges, along with a short glossary of BASIC commands, is also included.

Graphics programming isn't difficult if it's taught in a friendly fashion. SOUND AND GRAPHICS teaches graphics in colorful and musical ways!

RESTON PUBLISHING COMPANY, INC.
A Prentice-Hall Company
Reston, Virginia

ISBN 0-8359-7055-8

DATAMOST
20660 Nordhoff Street, Chatsworth, CA 91311-6152
(818) 709-1202

0 48831 00156

SOUND AND GRAPHICS

Addison Publishing Company Inc.